



**INSTITUTO POLITECNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO**

CENTRO DE INVESTIGACIONES ECONÓMICAS, ADMINISTRATIVAS Y SOCIALES

**“Caracterización del proceso de desarrollo del Software Libre y de Fuente Abierta
mediante el modelo de innovación abierta”**

TESIS

PARA OBTENER EL GRADO DE:

MAESTRO EN POLÍTICA Y GESTIÓN DEL CAMBIO TECNOLÓGICO

PRESENTA:

ALDO LIMA RAMOS

DIRECTORES DE TESIS:

DR. RUBEN OLIVER ESPINOZA

DR. SERGIO ORDOÑEZ GUTIERREZ

CIUDAD DE MÉXICO

DICIEMBRE DE 2016



INSTITUTO POLITÉCNICO NACIONAL SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México siendo las 12:00 horas del día 25 del mes de noviembre del 2016 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación de CIECAS para examinar la tesis titulada:
Caracterización del proceso de desarrollo del software libre y de fuente abierta mediante el modelo de innovación abierta

Presentada por el alumno:

Lima	Ramos	Aldo
Apellido paterno	Apellido materno	Nombre(s)
		Con registro: B 1 4 0 2 0 7

Maestría en Política y Gestión del Cambio Tecnológico

Después de intercambiar opiniones los miembros de la Comisión manifestaron **APROBAR LA TESIS**, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de tesis

Dr. Rubén Oliver Espinoza

Dr. Sergio Ordoñez Gutiérrez

Dr. Humberto Merritt Tapia

Dra. Hortensia Gómez Viquez

M. en P. Juan Carlos Becerril Elias

PRESIDENTE DEL COLEGIO DE PROFESORES

Dra. Gabriela María Luisa Riquelme Alcantara



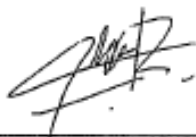
SECRETARÍA DE EDUCACIÓN PÚBLICA
INSTITUTO POLITÉCNICO NACIONAL
CENTRO DE INVESTIGACIONES
ECONÓMICAS ADMINISTRATIVAS

Instituto Politécnico Nacional
Secretaria de Investigación y Posgrado

CARTA DE CESIÓN DE DERECHOS

En la Ciudad de México, el día 18 del mes de octubre del año 2016, el que suscribe, Aldo Lima Ramos, alumno del Programa de Maestría en Política y Gestión del Cambio Tecnológico, con número de registro B140207, adscrito al Centro de Investigaciones Económicas, Administrativas y Sociales, manifiesto que es el autor intelectual del presente trabajo de Tesis bajo la dirección de los doctores Rubén Oliver Espinoza y Sergio Ordoñez Gutiérrez y cede los derechos del trabajo titulado Caracterización del proceso de desarrollo de software libre y de fuente abierta mediante el modelo de innovación abierta, al Instituto Politécnico Nacional para su difusión , con fines académicos y de investigación.

Los usuarios de la información de deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o directores del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección aldolimaramos@gmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.



Aldo Lima Ramos

DEDICATORIAS

Esta tesis va dedicada a mis padres que me han apoyado para perseguir mis sueños, Gerardo Lima y Francisca Ramos; a mis hermanos que me han motivado a esforzarme, Gerardo y Delia Betzabé; y a mis sobrinos que me han enseñado a mantener viva a la curiosidad y a seguir aprendiendo cada día, Luis Felipe, Abraham y Héctor Santiago.

También dedicó esta investigación a los inventores que crean nuevas soluciones, a los emprendedores que buscan resolver problemáticas sociales y a las personas que aportan su tiempo y trabajo para construir un bien común.

AGRADECIMIENTOS

Agradezco al **Consejo Nacional de Ciencia y Tecnología (CONACYT)**, al Instituto Politécnico Nacional (IPN) y al Centro de Investigaciones Económicas, Administrativas y Sociales (CIECAS) por los apoyos para estudiar la Maestría en Política y Gestión del Cambio Tecnológico (MPyGCT) y para realizar esta tesis.

A mis directores de tesis, el Dr. Sergio Ordoñez, de CIEc-UNAM, y el Dr. Rubén Oliver del CIECAS-IPN por su guía y consejos.

A mi lectora de tesis, profesora y subdirectora de Investigación, Dra. Hortensia Gómez.

A mi lector de tesis, profesor y coordinador de la maestría, Dr. Humberto Merritt.

A mis profesores de la maestría Katya Luna, Pilar Monserrat Pérez, Federico Stezano, Rolando Jiménez, Adolfo Sánchez y Juan Carlos Becerril.

Al coordinador editorial de la Secretaria Académica del IPN, Dr. Xicoténcatl Martínez Ruiz y al maestro Noel Angulo.

A mis compañeros de maestría: Alexandra Ortiz, Janeth Galván, Juno Báez, Patricia Chávez, Eder Hernández, Irving Muñoz, Omar Márquez y Rodrigo Díaz. Así como a otros compañeros que conocí en la maestría: Enrique Ruíz, Fabio López, Diana Estrella, Florencia Mackenzie, Jessica Zúñiga, Sandra Espinosa, Yessica Nuñez, Daniel Granillo, Iván Valdés y Luis Zea.

A los exdirectores de la Facultad de Ciencias de la Electrónica de la BUAP, el M.C Fernando Porras Sánchez y el Dr. Jaime Cid Monjaras que fueron grandes mentores para mí y me guiaron para tomar el camino de la gestión tecnológica.

A compañeros de la gestión tecnológica: Arturo Broca de Tekugo; Marisol Sánchez, Enrique Guzmán y Juan Jesús González, de Rubik IE.

A Emilio Saldaña representante de Creative Commons México, Miguel Salazar de Codeando México, Luis Royero y Rafael Aguilar de Hacking Health, Cinthia Flores de TIAN- taller del Hábitat, Zaid Badwan de Mediprint, Adriana Molina de Agile Academy, Lucia Fernández de OuiShare, Kenza Zouaoui de SenseCube, entre otras muchas comunidades.

A personas que me apoyaron directa o indirectamente, como mis amigos que estudiaron, trabajaron o compartieron momentos significativos conmigo. A Evaristo Espinosa, Ricardo Quintero, Nancy Lima y a la señora Teresa Bautista.

Finalmente agradezco a mi familia por todo su apoyo: mis papás, Gerardo y Francisca; mis hermanos, Gerardo y Delia Betzabé; y mis sobrinos Luis Felipe, Abraham y Héctor Santiago.

Contenido

ACTA DE REVISIÓN DE TESIS.....	2
CARTA DE CESIÓN DE DERECHOS	3
DEDICATORIAS.....	4
AGRADECIMIENTOS	5
Contenido.....	6
Índice de ilustraciones	8
Índice de tablas.....	9
Índice de acrónimos y abreviaturas	10
Glosario	11
RESUMEN	12
ABSTRACT.....	12
Introducción	13
CAPITULO 1: ENTORNO Y CARACTERISTICAS DEL SOFTWARE	17
1.1 La industria electrónica y la industria de cómputo	17
1.2 Industria del software	23
1.3 Características del Software	28
1.4 Software Libre y de fuente abierta (SLFA)	32
1.5 Propiedad Intelectual y licencias de software	37
CAPITULO 2: PROCESO DE INNOVACIÓN DEL SOFTWARE LIBRE Y DE FUENTE ABIERTA.....	41
2.1 Conceptos de Innovación y conocimiento.....	41
2.2 Modelos de gestión de la innovación.....	47
2.3 Innovación colaborativa y gestión de la innovación en el software libre y de fuente abierta	56
2.4 Agentes involucrados en el proceso de desarrollo del software libre y de Fuente abierta ...	58

2.5 Modelos de desarrollo de software libre y de software de fuente abierta.....	63
CAPITULO 3: CARACTERISTICAS DEL PROCESO DE DESARROLLO DE SOFTWARE LIBRE Y DE FUENTE ABIERTA.....	66
3.1 Entradas del modelo de innovación abierta en el proceso de desarrollo de software libre y de fuente abierta	66
3.2 Salidas del modelo de innovación abierta del proceso de desarrollo de software libre y de fuente abierta	73
3.3 Proceso de aprendizaje. Imitación- innovación	78
3.4 Gestión de la colaboración en el proceso de desarrollo de software libre y de fuente abierta	83
3.5 Aprendizajes y expectativas del proceso de desarrollo del software libre y de fuentes abierta	87
CONCLUSIONES.....	90
Referencias.....	94
ANEXO 1: Ciclo de sobre expectativa de tecnologías emergentes	99
ANEXO 2: Licencias de software libre y de fuente abierta	100
ANEXO 3: Evolución de las distribuciones de GNU/Linux	102

Índice de ilustraciones

Ilustración 1: Primeros diagramas de estructura de computadoras	18
Ilustración 2: Transformación del paradigma de uso de computadoras	21
Ilustración 3: Esquema de categorías de software	31
Ilustración 4: Logo de Free Software Foundation.....	33
Ilustración 5: Logo de Open Source Initiative.....	34
Ilustración 6: Compatibilidad de licencias.....	39
Ilustración 7: Semáforo de licencias Creative Commons	40
Ilustración 8: Grado de novedad de las innovaciones	42
Ilustración 9: Matriz de la innovación	44
Ilustración 10: Datos, información y conocimiento.....	45
Ilustración 11: Proceso de conversión de tecnología	46
Ilustración 12: Modelo lineal de innovación empujado por la tecnología.....	48
Ilustración 13: Modelo lineal de innovación atraída por la demanda	48
Ilustración 14: Modelo de innovación mezclado.....	49
Ilustración 15: Modelo de innovación integrado con funciones paralelas	50
Ilustración 16: Modelo de innovación integrado con retroalimentación	50
Ilustración 17: Curvas de costos y tiempo de desarrollo	51
Ilustración 18: Matriz de condiciones del proceso y resultado de la innovación.....	52
Ilustración 19: Enfoque de estrategias del modelo de gestión de la innovación	53
Ilustración 20: Modelo de innovación abierta.....	55
Ilustración 21: Innovación abierta de entrada	55
Ilustración 22: Innovación abierta de salida.....	56
Ilustración 23: Evolución del modelo de innovación abierta	57
Ilustración 24: Desarrolladores de Software libre y de fuente abierta.....	59
Ilustración 25: Proceso de desarrollo de software libre	64
Ilustración 26: Proceso de desarrollo del software de fuente abierta	64
Ilustración 27: Flujos de conocimiento en tres modelos de I+D de software	67
Ilustración 28: Entradas en el proceso de desarrollo del software	68
Ilustración 29: Lanzamiento de la versión Beta	73
Ilustración 30: Pivoteo de solución de software	74
Ilustración 31: Salidas del modelo de innovación abierta.....	75
Ilustración 32: Versiones posteriores del Software de Fuente Abierta	76
Ilustración 33: Versiones posteriores del Software libre	77
Ilustración 34: Modelo de espiral práctica.....	81
Ilustración 35: Proceso de imitación-innovación.....	82
Ilustración 36: Evolución de las modelos organizacionales	84
Ilustración 37: Proceso de desarrollo del Proyecto Mozilla.....	86

Índice de tablas

Tabla 1: Características evolutivas de la industria del software.....	24
Tabla 2: Principales software de servidores web	28
Tabla 3: Tipos de software de acuerdo a sus funciones generales.....	29
Tabla 4: Características de las tres tipos de software de acuerdo a sus funciones	30
Tabla 5: Libertades del Software Libre.....	34
Tabla 6: Decálogo de Software de Fuente Abierta	35
Tabla 7: Comparación de características de software libre y software de fuente abierta	36
Tabla 8: Comparativo de modelos de gestión de la innovación	57
Tabla 9: Edades de desarrolladores de SLFA	60
Tabla 10: Repositorios de software públicos.....	70
Tabla 11: Plataformas de financiamiento colectivo.....	72

Índice de gráficas

Gráfica 1: Comparativo de la Ley de Moore teórico con resultados empíricos	19
Gráfica 2: Distribución de los tipos de software en sistemas operativos de computadoras	26
Gráfica 3: Distribución de mercado de sistemas operativos de Mac y Linux	26
Gráfica 4: Distribución del software en servidores web	27
Gráfica 5: Distribución de software privativo contra software libre y de fuente abierta en distintos entornos	28
Gráfica 6: Número de patentes otorgadas por la USPTO.....	38

Índice de acrónimos y abreviaturas

- SL Software Libre
- SFA Software de Fuente Abierta
- SLFA Software Libre y de Fuente Abierta (FOSS- Free and Open Source Software)
- FOSS Free and Open Source Software
- GNU GNU No es Unix (GNU- GNU's Not Unix)
- TIC Tecnologías de la Información y Comunicación
- SO Sistema operativo
- PC Computadora Personal (Personal Computer)
- FSL Fundación de Software Libre (FSF- Free Software Foundation)
- GPL General Public Licence
- I+D Investigación y desarrollo
- PI Propiedad Intelectual
- OMPI Organización Mundial de la Propiedad Intelectual
- FS Free Software (Software Libre)
- OSS Open Source Software

Glosario

- **Base de Datos:** Un conjunto de datos interrelacionados a menudo con una redundancia controlada, organizada de acuerdo con un esquema para dar servicio a una o más aplicaciones: los datos se almacenan de forma que puedan ser utilizados por programas diferentes sin conocimiento de la organización o la estructura de los datos. Se emplea un método común para añadir datos nuevos y para modificar y recuperar datos existentes.
- **Código fuente** son instrucciones de código que muestra cómo trabaja el programa.
- **Desconferencia**, también llamado No-Congreso o conferencia de espacio abierto, es una conferencia en la que los propios participantes y asistentes toman un papel más participativo y activo.
- **Multiprocesador:** Un computador que tiene dos o más procesadores que disponen de acceso común a un almacenamiento principal.
- **Proceso:** Un programa en ejecución. Un proceso es controlado y planificado por el sistema operativo, es lo mismo que una tarea.
- **Servidor:** En una red, una estación de datos que proporciona servicios a otras estaciones: por ejemplo, un servidor de archivos, un servidor de impresión o un servidor de correo.
- **Software:** Conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas
- **Sistema operativo:** Software que controla la ejecución de programas y ofrece servicios tales como la asignación de recursos, la planificación, el control de la entrada/salida y la gestión de los datos.
- **Tubo de vacío**, válvula de vacío, válvula electrónica, válvula termoiónica o simplemente bulbo son los nombres del componente electrónicos, empleados para procesos de las primeras computadoras y permitió el desarrollo de la electrónica. Es antecesor al transistor.
- **Wearables** son prendas de vestir o accesorios a los que se les incorporo dispositivos tecnológicos para realizar funciones inteligentes, principalmente monitoreo.

RESUMEN

El desarrollo del software se desenvuelve en un entorno altamente dinámico, por lo que adquiere nuevas estrategias y comportamientos, particularmente el desarrollo de software libre y de fuente abierta que se caracteriza por compartir el código fuente o conocimiento codificado y desarrollar colaborativamente los proyectos. Por ello, resulta importante analizar las características del proceso de desarrollo del software libre y del software de fuente abierta, empleando el modelo de innovación abierta, para identificar las buenas practicas que se podrían implementar en otros entornos.

ABSTRACT

Software development takes place in a highly dynamic environment so it acquires new behaviors and strategies, particularly the development of free and open source software that is characterized by sharing the source code or codified knowledge and collaboratively developing the projects. Therefore, it is important to analyze the characteristics of the process of development of free software and open source software, using the open innovation model, to identify good practices that could be implemented in other environments.

Introducción

“La web es más una creación social que técnica. La diseñé para un efecto social — para ayudar a las personas que trabajan juntas— y no como un juguete técnico. El objetivo último de la web es apoyar y mejorar nuestra existencia en la telaraña mundial. Nos agrupamos en familias, asociaciones y empresas. Desarrollamos la confianza a grandes distancias, y la desconfianza a la vuelta de la esquina.”

TIM BERNERS-LEE, Creador de la www (World Wide Web)

Las Tecnologías de la Información y Comunicación (TIC) han creado costumbres, hábitos, modas y tendencias en prácticamente todos los aspectos de la sociedad. Las TIC son poderosas herramientas tecnológicas que pueden emplearse como catalizador del desarrollo social y económico del mundo, por ello deben gestionarse adecuadamente para construir un futuro que mejore nuestra existencia.

La adopción de las TIC es un factor clave para el desarrollo estratégico de las empresas y organizaciones, particularmente las soluciones relacionadas con aplicaciones de software, ya que les otorga ventajas competitivas sobre sus competidores que no adoptan nuevas tecnologías. Las características del software permiten resolver problemáticas que difícilmente podrían ser resueltas por otros medios.

El software se caracteriza por ser un elemento intangible inmerso en dispositivos electrónicos con capacidad de procesamiento que proporciona soluciones tangibles en diversos entornos, por lo que es considerado un eslabón importante en la cadena productiva de muchas empresas y catalizador de desarrollo económico. Se encuentra presente en gran cantidad de dispositivos electrónicos como computadoras, teléfonos, automóviles, electrodomésticos, maquinaria, entre muchos otros.

La importancia del software se ha limitado al sector productivo, pero ha impactado en todos los entornos de la sociedad de distintas formas y actualmente hay una alta demanda de soluciones de software, por lo que las estrategias de desarrollo comúnmente utilizadas requieren evolucionar para afrontar los nuevos retos. En este sentido, se requiere analizar el software y su proceso de desarrollo, particularmente alternativas de desarrollo como lo son el software libre y el software de fuente abierta que implican modelos de desarrollo colaborativo en la que numerosas personas

aportan su trabajo y que comparten el código fuente para que otros puedan utilizar, entender y mejorar los proyectos.

Por tal razón, el objetivo de esta investigación es analizar el proceso de desarrollo de Software Libre y de Fuente Abierta, para reconocer las ventajas del modelo de desarrollo colaborativo que emplean. Por lo que surge las siguientes preguntas de investigación, que deberán responderse en la tesis: qué es lo que caracteriza al proceso de desarrollo, qué estrategias o comportamientos se pueden destacar en el proceso de desarrollo, cómo se lleva a cabo el proceso de desarrollo y cómo se gestiona la colaboración en proyectos de software libre y de fuente abierta, entre otras preguntas.

De antemano se propone como hipótesis que el desarrollo del Software Libre y del Software de Fuente Abierta implica un nuevo sistema de colaboración basado en compartir el código fuente de sus programas que, por un lado, acelera el proceso de desarrollo de software y, por el otro, crea soluciones de software más eficientes, ya que se aprovecha el trabajo previo y las contribuciones de numerosas personas.

Por otro lado, la estrategia de innovación colaborativa que emplea el Software Libre y de Fuente Abierta que permite desarrollar grandes proyectos y mejorar las capacidades tecnológicas de los agentes involucrados en el proceso de desarrollo justifica el estudio de este tema de investigación, ya que de los procedimientos involucrados en el proceso de desarrollo colaborativo se pueden extraer aprendizajes y buenas prácticas que podrían ser replicados en entornos diferentes al software.

Metodología

El paradigma de desarrollo del software Libre y de Fuente Abierta se basa en la colaboración de una red de agentes que aportan su trabajo, por lo que resulta natural explicar la caracterización del proceso de desarrollo mediante el modelo de innovación abierta de Chesbrough.

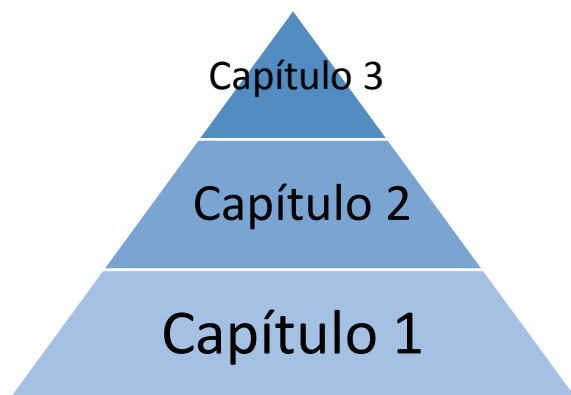
En primer lugar, se debe entender el contexto de la evolución del software, reconocer los sus diversos tipos y categorías, definir conceptos básicos del software, y describir las características

y condiciones actuales del Software Libre y del Software de Fuente Abierta. En segundo lugar, se requiere definir conceptos relacionados con la innovación y modelos de gestión de la innovación, así como los agentes y procedimientos implicados en el desarrollo de Software Libre y de Fuente Abierta. Finalmente, a partir de la construcción teórica anterior y mediante el modelo de innovación abierta, se debe examinar las características que mejor describen el proceso de desarrollo de Software Libre y de Fuente abierta.

En otras palabras, la caracterización del proceso de desarrollo del Software Libre y de Fuente Abierta se basa en la construcción de bases que permitan entender, por un lado, el contexto y características generales del software y, por el otro, describir el proceso de desarrollo del software empleando conceptos de innovación, conocimiento y el marco conceptual de la innovación abierta.

Estructura de la tesis

La tesis se ha dividido en tres capítulos, para abordar el tema de lo general a lo particular, del entorno que rodea al software, al proceso de desarrollo, para finalmente indagar en las características particulares que representa desarrollar software libre y de fuente abierta.



- Capítulo 1: Entorno y características del software.

Se introduce el contexto del software, iniciando por su origen dentro de la industria del cómputo y su proceso de independencia, para definir las características propias y tipos de software. Además de las clasificaciones del software, donde se introducen el software privativo, el software libre y el software de fuente abierta.

- Capítulo 2: Proceso de innovación del software libre y de fuente abierta

Se parte de definiciones de innovación, para posteriormente revisar la evolución de modelos de gestión de la innovación que refuerza la justificación de emplear el modelo de innovación abierta para analizar el proceso de innovación en el SLFA. El modelo de innovación abierta contempla la compleja y variada interacción con el exterior de la organización, tanto la entrada de tecnología externa, como la salida de tecnología que puede competir en mercados de otras empresas o incluso crear nuevos mercados y/o modelos de negocio.

- Capítulo 3: Características del proceso de desarrollo de software libre y de fuente abierta

Se describen las principales características del proceso de desarrollo de Software Libre y de Fuente Abierta, que es capaz de gestionar la participación y conocimiento de números actores mediante una red de colaboración para desarrollar soluciones de software. Además de indagar sobre el futuro del código abierto.

CAPITULO 1: ENTORNO Y CARACTERISTICAS DEL SOFTWARE

1.1 La industria electrónica y la industria de cómputo

Las Tecnologías de la Información y Comunicación (TIC) son sistemas tecnológicos basados en computación, informática, telecomunicaciones capaces de adquirir, almacenar, procesar y transmitir información (Sampedro, 2011), para facilitar la comunicación interactiva entre dos o más individuos en una red (RICYT, 2006) y estimular la generación, intercambio, difusión, gestión y acceso al conocimiento; y que han transformado drásticamente nuestros comportamientos, modificando o creando paradigmas, hábitos y costumbres en la sociedad (Cobo, 2009).

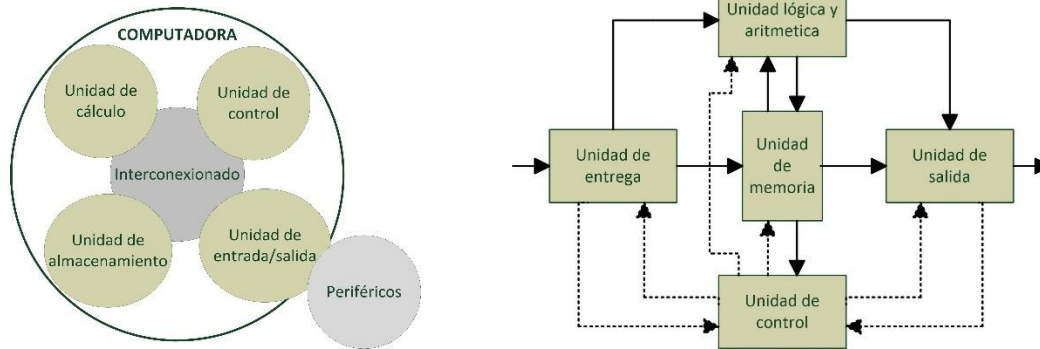
La relevancia de las TIC se puede apreciar cuando se compara con otras industrias. Las TIC, en el 2010, tuvieron un crecimiento anual de 5.7%, mientras que otras industrias importantes como la automotriz y de la construcción solo crecieron 2.6% y 2.4% respectivamente. Y dentro de los elementos que comprenden las TIC, destaca la industria del software que ha tenido una mayor inversión destinada a Investigación y Desarrollo (I+D) que es de aproximadamente 15%, muy superiores a la inversión de importantes industrias como la aeroespacial y automotriz que no superan el 9% (INCO, 2014).

El software es un elemento relativamente nuevo que por su naturaleza digital e intangible, requiere de un componente físico para existir, también conocido como hardware. Sus orígenes están ligados al surgimiento de la primera generación de computadoras totalmente electrónicas en la década de 1940, como Colossus o ENIAC (Joskowicz, 2016). La evolución del cómputo ha pasado por distintas fases, desde el uso de tarjetas perforadas, el funcionamiento a partir de bulbos, hasta los avances y estandarizaciones de la década de 1960 que posibilitaron el intercambio de periféricos y que dio origen al modelo de computadora más usual, la computadora personal (PC) (Ordoñez, 2004).

La computadora nace de esfuerzos por acelerar y automatizar el procesamiento de cálculos matemáticos. En 1930, algunos investigadores reconocían los componentes necesarios para el funcionamiento de una máquina calculadora automática, como unidad de cálculo, unidad de control, unidad de almacenamiento, unidad de entrada/salida, periféricos e interconexión. Posteriormente, en el desarrollo de ENIAC, la primera computadora electrónica de uso general, Von

Neuman propone un modelo y arquitectura de computadora, además de algunos postulados que serían la base de la industria de cómputo (Olivo, 1999).

Ilustración 1: Primeros diagramas de estructura de computadoras



a) Máquina calculadora automática

b) Arquitectura de computadora de Von Neumann

Fuente: Elaboración propia a partir de Olivo, 1999.

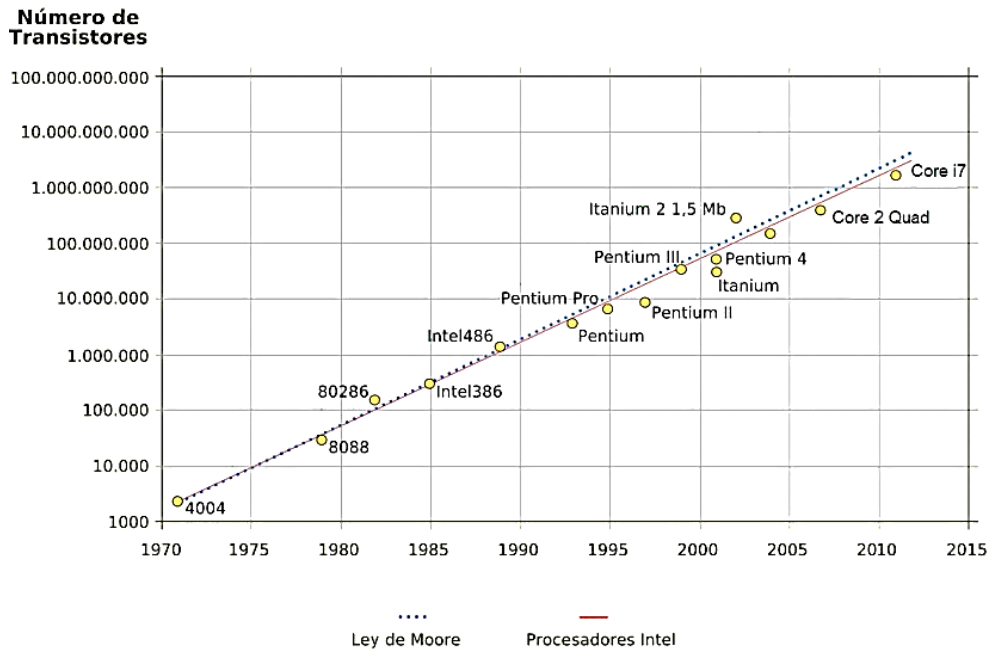
El progreso de la industria del cómputo ha dependido directamente de la evolución del hardware, que comprende desde periféricos, tarjeta madre, disco duro, memoria RAM, hasta el procesador (o microprocesador). El procesador es un elemento clave que se puede comparar o definir de manera simbólica como el cerebro de la computadora, ya que ejecuta los procesos e instrucciones. Por ello, el aumento de la capacidad de procesamiento de los procesadores se relaciona directamente el aumento de la potencia de las computadoras.

En 1965, Gordon Moore, cofundador de Intel, analizó la evolución de los procesadores de las computadoras y propuso una regla conocida como la Ley de Moore, que describe el crecimiento del número de transistores en un procesador. La Ley de Moore, postula que la potencia de los ordenadores aumentaría exponencialmente, es decir que el número de transistores en un área de un centímetro cuadrado de los procesadores se duplicaría cada año. Posteriormente, una década después Moore reajustó el periodo de tiempo de un año a 18 meses o menos, para que se adaptará mejor al comportamiento real del crecimiento de número de transistores en los procesadores (Cheang Wong, 2005; Lima, 2015).

Las predicciones de la ley de Moore se aproximan bastante a los valores reales del número de transistores por centímetro cuadrado de los procesadores que se han sacado al mercado desde 1970. En la Gráfica 1, se compara los valores teóricos de la ley contra los resultados prácticos del

número de transistores en los procesadores comerciales de Intel. En el eje horizontal se grafica un periodo de tiempo de 45 años, mientras que en el eje vertical muestra el número de transistores que tienen los procesadores, en escala logarítmica¹.

Gráfica 1: Comparativo de la Ley de Moore teórico con resultados empíricos



Fuente: guiahardware.es, 2015.

La predicción de la ley de Moore es representada por una recta de líneas punteadas, mientras que las condiciones reales del mercado se representan con puntos que simbolizan los distintos procesadores de Intel, ubicados de acuerdo a su año de lanzamiento comercial y a su número de transistores por centímetro cuadrado. La ubicación de los procesadores en la gráfica permite trazar una recta de la tendencia que existe en los procesadores comerciales, para permitir una mejor comparación de los resultados empíricos de los procesadores de Intel y la representación teórica de la ley de Moore., y observar que ambas líneas son prácticamente idénticas.

Esta correlación de los valores teóricos y prácticos de la ley de Moore sigue siendo alta, incluso décadas previas a la proclamación de la ley: mediados de la década de 1940, la computadora

¹ Escala logarítmica es una graduación numérica no lineal, que sirve para representar de mejor forma el crecimiento exponencial de una función.

ENIAC funcionaba a partir de 18,000 tubos de vacío², cuyo valor corresponde a un punto de la recta de valores teóricos de la ley de Moore extendida algunos años antes y también para el futuro, se espera que esta ley se siga cumpliendo al menos hasta el año 2025 (Kaku, 2014; Cheang Wong, 2005).

En este sentido, algunos ejemplos permiten evidenciar el progreso del cómputo, como: la potencia de procesamiento de las tarjetas musicales actuales es mayor que todo el equipo de cómputo de las fuerzas aliadas, en la Segunda Guerra Mundial; o los teléfonos móviles actuales que tienen mayor potencia de procesamiento que toda la NASA en 1969, cuando llevaron a los primeros astronautas a la Luna; y seguramente en las próximas décadas tendremos supercomputadoras o inteligencia artificial con capacidad de procesamiento inimaginable al alcance de cualquier persona (Kaku, 2014; Lima, 2015).

En las primeras décadas de la computadora electrónica, su uso estaba restringido a algunos centros de investigación, principalmente militares y se ocupaban entre muchas personas, debido a las dificultades técnicas y costos implicados. Posteriormente los avances tecnológicos permitieron que las computadoras se introdujeran en el mercado con costos y tamaños considerablemente menores, lo que supuso que en algunos sectores de la sociedad se llegara a tener una computadora por persona. La introducción de la red de redes, conocida como internet, posibilitó la interconexión de las computadoras para compartir recursos e información, para ampliar la capacidad de procesamiento y almacenamiento del que podía disponer un usuario con computadora.

Actualmente, la evolución técnica de las computadoras ha impactado directamente en una continua disminución en su tamaño y precio, que se refleja en la diversificación de su oferta, tales como sistemas mínimos de computadora que resuelven soluciones específicas empleando software. Algunos ejemplos de dichos sistemas mínimos son los *wearables*³ como pulseras que miden el ritmo cardiaco y signos vitales; o refrigeradores que son capaces de alertar y solicitar algún producto cuando este por terminarse.

² Tubo de vacío, válvula de vacío, válvula electrónica, válvula termoiónica o simplemente bulbo son los nombres del componente electrónicos, empleados para procesos de las primeras computadoras y permitió el desarrollo de la electrónica. Es antecesor al transistor.

³ Wearables son prendas de vestir o accesorios a los que se les incorporo dispositivos tecnológicos para realizar funciones inteligentes, principalmente monitoreo.

El progreso del cómputo ha transformado el paradigma del uso de la computadora, en las primeras décadas del surgimiento de la computadora, había muchas personas alrededor de una computadora, posteriormente en algunos entornos se tenía una computadora por persona, y en este momento se llega a tener muchas computadoras a nuestro alrededor.

Ilustración 2: Transformación del paradigma de uso de computadoras



Fuente: Elaboración propia.

En esta última etapa del proceso de transformación del paradigma de uso de computadora, las personas cuentan con numerosas computadoras a su alrededor, aunque en una versión compacta y de aplicación específica. Estas computadoras a la medida se encuentran presentes en prácticamente cualquier aparato electrónico: electrodomésticos, vehículos, dispositivos móviles; conocidos como sistemas mínimos de computadora, ya que se componen de elementos básicos de hardware y software indispensables para cumplir con una función (Lima Ramos, 2015). También pueden considerarse como computadoras a FPGA, Arduino, Raspberry PI, ya que son plataformas electrónicas programables.

Pronto, será más común emplear términos relacionados con las tecnologías o dispositivos inteligentes integrados a nuestro alrededor, por lo que el término de computadora se irá difuminando y habrá la necesidad de redefinir este concepto (Kaku, 2014; Lima Ramos, 2015). Puesto que el hardware es la base del software, es necesario conocer las principales tendencias tecnológicas relacionadas con la industria electrónica y del software.

Después de la crisis financiera de 2007 a 2009, las tendencias tecnológicas se reajustaron, decayendo algunas y surgiendo otras. Gartner Inc. analiza y clasifica periódicamente las tendencias tecnológicas emergentes, además evalúa su madurez y tiempo de adopción masiva de estas

tecnologías (Ver Anexo 1). Las tendencias actuales pueden ser agrupadas en tres tendencias generales (Levy, 2015; Ordoñez, en prensa).

La primera tendencia se refiere a la combinación del mundo físico y el mundo virtual, es decir una combinación de ambientes físicos, virtuales y electrónicos, mediante una red de dispositivos que proporcionan una nueva experiencia al usuario, empleando dispositivos móviles, wearables, dispositivos electrónicos del hogar, realidad aumentada, realidad virtual, materiales de impresión 3D y otras tecnologías.

La segunda tendencia aborda el desarrollo de las máquinas inteligentes, mediante la generación y transmisión de información generada por una red de dispositivos, por lo que surge la necesidad de crear máquinas de aprendizaje avanzado que interpreten los datos y puedan aprender del ambiente, como son los asistentes personales virtuales, vehículos autónomos y otros dispositivos autónomos.

La tercera tendencia se basa en el desarrollo de nuevas arquitecturas y plataformas de hardware y software, ya que para desarrollar nuevas soluciones con dispositivos inteligentes se requieren arquitecturas de sistemas avanzados de soporte que tengan cierta flexibilidad para la creación de aplicaciones basadas en software, como FPGAs, Arduino, Raspberry PI, PLCs y otras plataformas electrónicas que soportan nuevas soluciones como los dispositivos del internet de las cosas.

Dentro de estas tendencias tecnológicas, el software es un factor clave, que está creando una sobredemanda de aplicaciones de software, que requerirán de un gran número de especialistas de cómputo, particularmente programadores (code.org, 2014). Aunque el progreso técnico del hardware se detuviera, el software seguiría evolucionando, ya que la ley de Moore llegará a su fin para la próxima década y la evolución de los procesadores se detendrá⁴.

Sin embargo actualmente se está desarrollando lo que será la siguiente revolución tecnológica, basada en un nuevo sistema de cómputo conocida como computación cuántica. La computación cuántica dará lugar a un nuevo paradigma de procesamiento de información, para el

⁴ Aunque la Ley de Moore llegué a su fin, seguirán surgiendo nuevas arquitecturas que mejoren el desempeño de los procesadores.

cual se tendrían que crear nuevos modelos y métodos de programación antes no conocidos (Vélez, Sicard, 2000).

La primera diferencia de la computación cuántica radica en condiciones de 4 estados, en lugar de dos, que aumentará exponencialmente las posibilidades y capacidad de procesamiento. Con este nuevo paradigma de cómputo tendríamos la capacidad de procesamiento para resolver problemáticas y condiciones sumamente complejas que actualmente son imposibles de solucionar. La computación cuántica ha dado sus primeros pasos, IBM ha ofrecido recientemente sus servicios de cómputo cuántico en la nube, llamada *IBM Quantum Experience*, cuyo procesador cuántico está compuesto de cinco qubits superconductores, sin embargo se espera que en la próxima década el tamaño promedio de los procesadores cuánticos será de 50 a 100 qubits (IBM, 2016a y 2016b).

1.2 Industria del software

La industria del software surge a partir de la industria del cómputo, a mediados del siglo pasado, su evolución está ligada al desarrollo del hardware, sin embargo la trascendencia del software lo volvió mucho más valioso que el hardware de las computadoras. Se transformó de un complemento intangible de las computadoras a un producto/servicio que ofrece soluciones tecnológicas y que ha creado un sólido ecosistema económico a su alrededor.

La evolución del software es explicada mediante 6 etapas principales, descritas en la Tabla 4, desde el momento de la creación de la industria hasta las condiciones actuales (Sampedro, 2011). Después del surgimiento del internet como la red de redes, se revolucionó la industria del software, por lo que se podría hacer una nueva clarificación a partir de ese momento.

En las etapas iniciales de la industria del software: *Gestación y Nacimiento*: Los primeros desarrollos de software fueron para aplicaciones militares y científicas, las computadoras eran vendidas sin software, solo en algunos casos su producción era llevada a cabo por los mismos fabricantes de computadoras, además para cada computadora se le desarrollaba software específico de acuerdo a sus requerimientos de uso.

Los primeros intentos por simplificar el proceso de programación surgen desde el desarrollo de la primera computadora de propósito general, ENIAC, con el matemático Jonh Von Neuman que

propuso las instrucciones que podrían almacenarse en la propia computadora, lo que sentó las bases para la creación del sistema operativo de las computadoras, lo que posteriormente facilitaría la ejecución de aplicaciones sobre la plataforma del sistema operativo. En los años 50s, los usuarios comenzaron a desarrollar sus propias aplicaciones de software, pero fue hasta IBM estandariza el sistema operativo de la familia de computadoras S/360, que la venta del software se independiza del hardware y a partir de ese momento nace la industria del software.

Tabla 1: Características evolutivas de la industria del software⁵

Etapa	Características
1ª Gestación 1940-1950	El sw es desarrollado por los productores de hw. El sw es específico para cada usuario, es artesanal con aplicaciones científicas y militares. El uso industrial de hw estimuló el desarrollo de sw. Los usuarios cooperaban en el desarrollo de mejores aplicaciones.
2ª Nacimiento Década de 1960	Con la IBM/360 introducida en 1964, el sw (SO único) se hace compatible en diferentes tamaños de computadora. El sw es desarrollado por empresas independientes de la producción de hw. 1969 IBM separa los precios de software del hardware.
3ª Crecimiento Década de 1970	A principios de los 70s, en EU había entre 1500 y 2800 empresas. El sw empacado se hace masivo para las PC y se acelera el crecimiento de ventas. En 1971 IBM introduce el hard disk que permite almacenar grandes cantidades de información, mejora la arquitectura de hw y diversifica las actividades de sw. Problemas de calidad, administración y medición.
4ª Consolidación Década de 1980	Se consolidan las empresas independientes de sw; Microsoft, Lotus, Wordperfect, Ashton-Tate, Borland, entre otras. Bajas barreras de entrada. Diseño dominante: PC. Homogenización de sistemas operativos y aplicaciones para PC.
5ª Uso de "redes" Finales de 1980	Surge Intranet e Internet basados en la plataforma PC.
6ª Sistemas abiertos Década de 1990-?	Inicia el desarrollo de sw de fuente abierta en comunidades virtuales de desarrolladores.
7ª Computación ubicua Posterior al 2007-?	Es parte de una evolución constante de la tecnología, pero tiene un mayor crecimiento después de la crisis económica del 2007. Esta etapa se refiere al aumento de dispositivos electrónicos que se establecen a nuestro alrededor, como un smog de tecnología. Es una etapa con mayor demanda de aplicaciones.

Fuente: Elaboración propia basada en Sampredo, 2011.

En las siguientes etapas de desarrollo de la industria, Crecimiento y Consolidación: Las empresas de software independientes conformaron la industria del software, que tuvo gran crecimiento en la productividad en la década de 1970, debido a la introducción del disco duro y la

⁵ Sw: software, hw: hardware

mejora del desempeño del hardware lo que permitió la consolidación de varias empresas. La creación de redes entre empresas y usuarios fue un factor clave para su desarrollo y expansión, sin embargo la industria enfrentó problemas debido a los altos costos de mantenimiento y administración en algunas aplicaciones (Sommerville, 2005).

En estos periodos prevaleció el software hecho a la medida, y la PC se convirtió en el diseño de computadora dominante, que permitió estandarizar sistemas operativos y aplicaciones. Además nacieron grandes empresas pioneras en la industria de la computación y software como Apple Computer Inc., Microsoft, Borland, Lotus entre otras, que se basaban en desarrollos tecnológicos intangibles, como el software, volviéndose altamente rentables. Actualmente estas empresas tienen ingresos anuales de millones de dólares (FORBES, 2015).

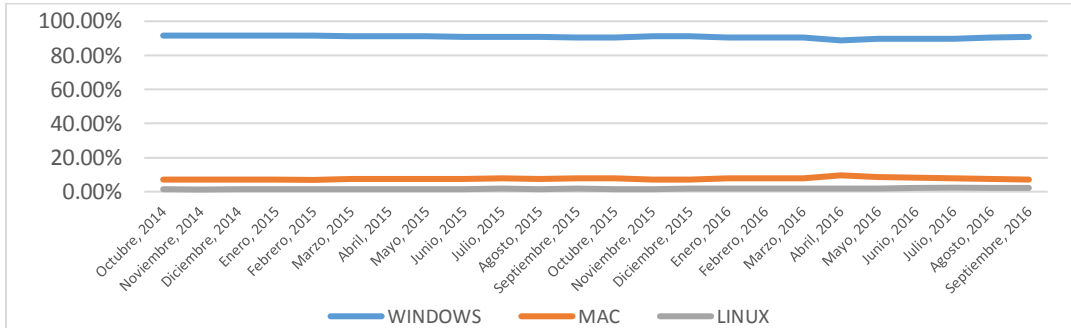
Finalmente en las etapas de creación de redes, Uso de redes y Sistemas abiertos: En las organizaciones como empresas e instituciones públicas, el estándar de la PC posibilitó la creación de redes internas como intranets, que posteriormente sentarían la base del internet. El desarrollo del software fue impulsado por el crecimiento de las capacidades de las redes. El internet permitió a los usuarios desarrolladores compartir sus programas y códigos fuente, lo que dio origen a movimientos como el de software libre, basados en el trabajo colaborativo y en compartir el código fuente del software.

Actualmente nos encontramos en la 6ª etapa, con un auge de la industria del software con un incremento de la capacidad de procesamiento de información, junto con el aumento de accesibilidad y difusión de la información y el conocimiento (Ordoñez, 2007). El software libre y el software de fuente abierta han sido un factor clave para la construcción del ecosistema digital actual. De acuerdo a la revisión del estado de la técnica, podría proponer una 7ª etapa: computación ubicua, descrita con una mayor demanda aplicaciones de software en diversos dispositivos a nuestro alrededor, fenómeno también conocido como el internet de las cosas, que tuvo mayor impacto después de la crisis económica global de 2007 a 2009 (Ordoñez, en imprenta).

El software libre y de fuente abierta ha sido menospreciado o considerado insignificante, seguramente se debe que el software más común es el sistema operativo de las computadoras casi en su totalidad software privativo de la empresa Microsoft. Los sistemas operativos de

Microsoft tienen más del 91% del mercado, mientras que las distribuciones de Linux (Software libre) apenas superan el 2% (NETMARKETSHARE, 2016).

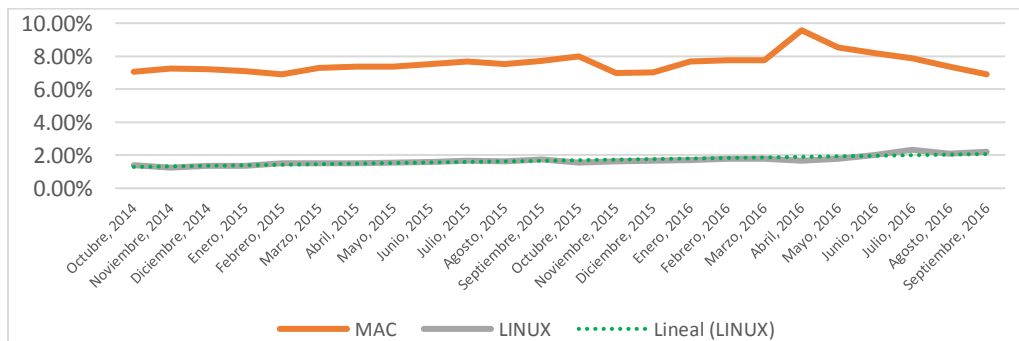
Gráfica 2: Distribución de los tipos de software en sistemas operativos de computadoras



Fuente: Elaboración propia basada en los datos de Netmarketshare, 2016.

En la Gráfica anterior, se observa el dominio casi monopolístico de los sistemas operativos de Microsoft que ronda por 90% en el periodo de dos años de la muestra, de octubre de 2014 a septiembre de 2016. Sin embargo en una revisión a más detalle de los sistemas operativos de Mac y Linux, en la siguiente Gráfica, se observa que la cuota de mercado de Mac oscila entre 7% y 8%, aunque en abril del 2016 tiene un pico que casi llega al 10% y posterior a ello tiene una caída por debajo del 7%. En el caso de Linux, se observa un crecimiento mantenido diminuto, recientemente ha superado 2% de la distribución y manteniendo la misma tendencia lineal se esperaría que en octubre de 2017 llegué a 2.5%.

Gráfica 3: Distribución de mercado de sistemas operativos de Mac y Linux

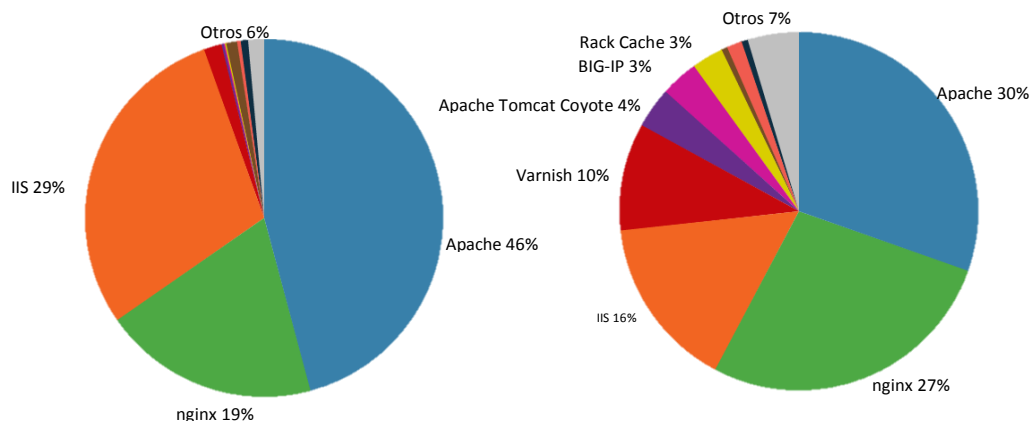


Fuente: Elaboración propia basada en los datos de Netmarketshare, 2016.

Sin embargo, en escenarios técnicos más especializados como en servidores web, existe un dominio por Apache, de software libre, con un 46%, ya que ISS (Internet Information Services), de software propietario, tiene un 29%, el porcentaje restante corresponde principalmente a servidores

web con otras versiones de software libre y de fuente abierta. Mientras que el software privativo tiene alrededor del 90% del mercado en sistemas operativos de computadoras, en servidores web apenas llega al 30%.

Gráfica 4: Distribución del software en servidores web



a) En todos los sitios de internet

b) En el top 10,000 de sitios de internet

Fuente: Elaboración propia basada en el sitio *builtwith*, 2016.

Por otro lado, en un análisis del software de los servidores web que utilizan el top de 10,000 sitios de internet, los porcentajes cambian considerablemente. ISS de Microsoft reduce su porcentaje de 29% a 16%, en el caso de Apache se reduce de 46% a 30%, mientras que los demás software aumentan su distribución (*built with*, 2016). Dentro de los primeros 10 sistemas operativos de servidores web, hay 7 de ellos con software libre o de fuente abierta, cuyo porcentaje aumenta a casi 80%.

Por otro lado, en sistemas operativos de dispositivos móviles, el porcentaje de software libre y de fuente abierta supera el 70%, mientras que en los bancos de Wall Street, 9 de cada 10 servidores trabajan con software libre o de fuente abierta (*Linux IT Europe*, 2011), mientras que el 94% del top 500 de superordenadores tienen software libre o de fuente abierta, que son escenarios que requieren mayor seguridad y robustez. Estas condiciones también se repiten en los centros o agencias de investigación como la NASA, donde han optado por usar Software libre o de fuente abierta, han instalado distribuciones de Linux tanto en las computadoras portátiles, en sus súper computadoras, incluso en un robot usado por los astronautas (*20 Minutos*, 2015).

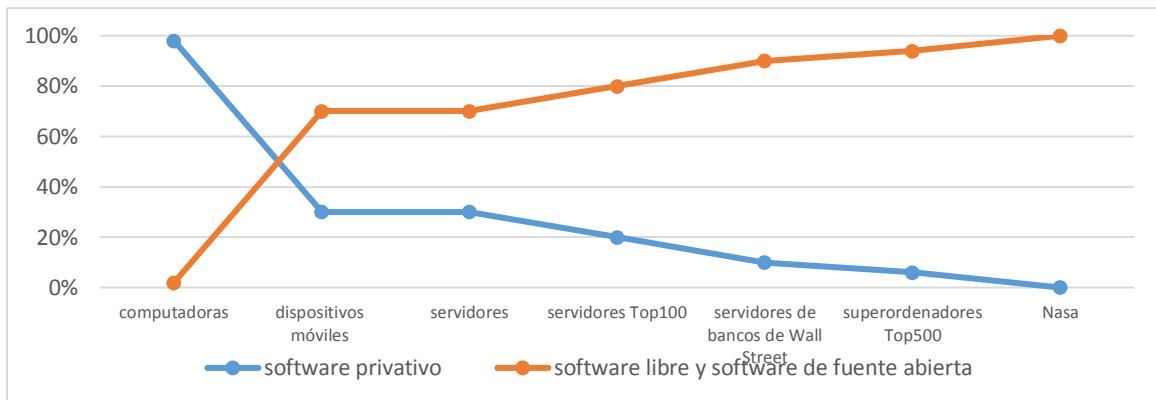
Tabla 2: Principales software de servidores web

#	NOMBRE DEL SOFTWARE	CLASIFICACIÓN DEL SOFTWARE
1	Apache	Software libre/ fuente abierta
2	Nginx	Software libre/ fuente abierta
3	IIS	Software propietario
4	Varnish	Software libre/ fuente abierta
5	Apache Tomcat Coyote	Software libre/ fuente abierta
6	BIG-IP	Software propietario
7	Rack Cache	Software libre/ fuente abierta
8	LiteSpeed	Software libre/ fuente abierta
9	Phusion Passenger	Software libre/ fuente abierta
10	Application Request Routing	Software propietario

Fuente: Elaboración propia basada en el sitio builtwith, 2016.

En la siguiente gráfica se muestra los porcentajes de la cuota del mercado que tiene tanto el software privativo como el software libre y de fuente abierta, donde se observa que el software propietario tiene un dominio casi total, sin embargo en otros escenarios más especializados la distribución se invierte. El software libre y de fuente abierta tiene mayor impacto en escenarios donde se requiere mayor seguridad y robustez.

Gráfica 5: Distribución de software privativo contra software libre y de fuente abierta en distintos entornos



Fuente: Elaboración propia

1.3 Características del Software

El software se puede definir como un paquete tecnológico intangible que describe un algoritmo mediante instrucciones codificadas y plasmadas en un programa. El programa está conformado por líneas de código de programación que ejecutan procesos, a lo que se le llama código

fuente. El código fuente es el núcleo del paquete tecnológico, es conocimiento codificado con aplicación inmediata que por su naturaleza digital puede acumularse y transferirse fácilmente (Ordoñez, 2009; OCDE, 2009; Sampedro, 2011; IEEE, 1993).

La importancia del software radica en su capacidad de solución de problemas en distintas condiciones e industrias, incluso se le considera una pieza importante en considerables procesos productivos de todo tipo de industrias. Las bajas barreras de entrada, los bajos costos de difusión de los productos y una alta rentabilidad han propiciado el crecimiento de la industria del software, aunque la misma naturaleza y características del software han sido fundamentales para ello (Ordoñez, 2009).

Existen diversas clasificaciones del software, la primera clasificación que se abordará se basa en sus niveles de función general en la computadora: el software de sistema, el software de programación y el software de aplicación. En la siguiente tabla se muestran algunos ejemplos de cada uno de los tipos de software.

Tabla 3: Tipos de software de acuerdo a sus funciones generales

Tipos de software	Ejemplos
Software de sistema	
Software de programación	
Software de aplicación	

Fuente: Elaboración propia

Estos tipos de software tienen características particulares y distintos niveles de uso como: el Software de sistema consta de elementos básicos que administran y gestionan los recursos físicos de la computadora; el Software de programación son herramientas para desarrollar programas a partir de lenguajes de programación y un conjunto de instrucciones lógicas; y el Software de aplicación realizan tareas específicas son conocidas como programas o aplicaciones (Ordoñez, 2004; informaticxp.net, 2015). La descripción de esta clasificación está detallada en la siguiente Tabla.

Tabla 4: Características de las tres tipos de software de acuerdo a sus funciones

Tipos de software	Definición	Clasificación/ agrupación	Ejemplos
Software de sistema	Son necesarios para el funcionamiento de la computadora a través de la administración de los recursos del hardware	Sistemas operativos, controladores de dispositivo, herramientas de diagnóstico, herramientas de corrección y optimización, servidores, utilidades	Windows, GNU/Linux, Mac OS, Mandriva, Solaris, BSD, MS_Dos, Apache; Android, Symbian
Software de programación	Son herramientas de aplicación para desarrollar nuevos programas, con base a lenguajes de programación y de los conocimientos de lógicos de los programadores	Editores de texto, compiladores, intérpretes, enlazadores, depuradores y Entornos de Desarrollo Integrados (IDE).	Pascal, Visual Basic, Fortran, Cobol; ensamblador, Ruby, Python, Java, JavaScript, PHP, jQuery; Oracle, circle; NetBean
Software de aplicación	Son aplicaciones enfocadas en realizar tareas específicas, se instalan en el software de sistema. Es el tipo de software que más abunda	Hojas de cálculo, procesadores de texto, editores, programas de comunicaciones y programas de diseño asistido por computadora.	Word, Excel, Photoshop, SolidWorks, Autocad, Instagram, Angry birds

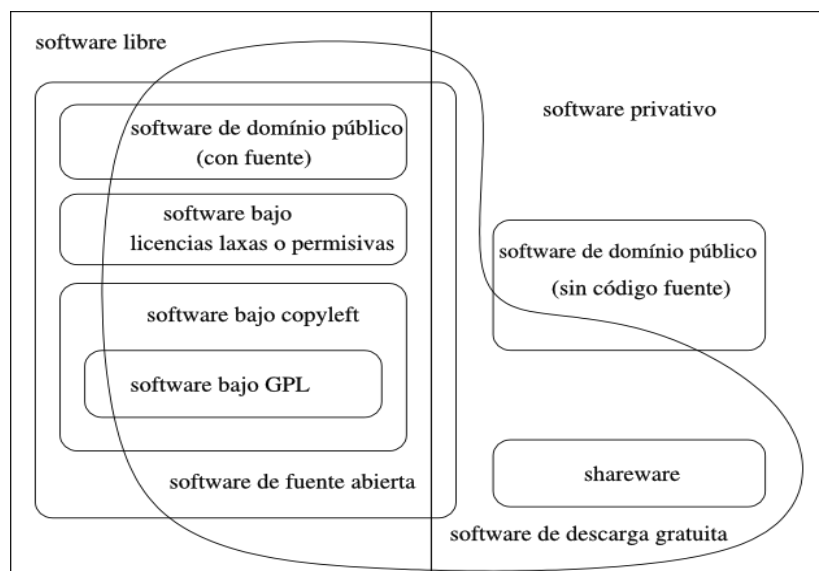
Fuente: Elaboración propia, a partir de: Ordoñez, 2004; informaticaxp.net, 2015

También existen otras posibles formas de clasificar el software como el propósito por el que fue diseñado, es decir si se diseñó para cumplir un propósito general para un gran número de usuarios, se dice que es un software empaquetado como Office Word, Autocad, Whatsapp, entre otros, por el contrario si el objetivo del software es cumplir necesidades particulares de algún cliente entonces se dice que es un software hecho a la medida como programas que gestionan procesos en alguna fábrica o aplicaciones solicitadas para algún uso específico con determinadas características.

Además hay una categoría particular, que es el software embebido, el cual permanece integrado en el hardware y a diferencia de otros tipos de software, el software embebido es vendido como parte del producto, por lo que debe ser muy confiable. Este tipo de software es la base de la computación ubicua y es usado en un amplio rango de dispositivos electrónicos (OCDE, 2009). Sin embargo, la investigación partirá respecto la siguiente clasificación que aborda las restricciones de uso del software, ya que determinan las formas en las que está permitido usar el software, desde las condiciones que limitan el uso hasta las que permiten el uso libremente del software.

Esta clasificación está basada en las definiciones de la Fundación de Software Libre. Algunos tipos de software son contrastados en un diagrama para distinguir las similitudes y diferencias, así como los conjuntos y subconjuntos que comprenden. En el universo completo del software representado por este esquema, cualquier software desarrollado se puede ubicar en algún punto. Los tipos de software incluidos son software privativo, software libre, software de fuente abierta, software de dominio público (con código fuente), software bajo licencias laxas o permisivas, software bajo copyleft, software bajo GPL, software de dominio público (sin código fuente), software de descarga gratuita y shareware (Ver Ilustración siguiente).

Ilustración 3: Esquema de categorías de software



Fuente: GNU, 2015.

En el esquema, se observa de manera general el comportamiento de algunos tipos de software. El primer software que destaca es el Software Libre, cuyas cualidades se describen bajo 4 leyes, también llamadas libertades, que debe cumplir obligatoriamente, para poder declararlo como Software Libre. La primera y tercera ley (libertad 0 y libertad 2) se refieren al permiso de uso y redistribución del programa, la segunda ley (libertad 1) autoriza su estudio y modificación del software, entretanto la cuarta ley (libertad 3) requiere que se distribuyan las modificaciones incluyendo el código fuente. En otras palabras un software libre siempre será libre, inclusive sus modificaciones, lo que posibilita una óptima y libre difusión del conocimiento (GNU, 2015).

Por otro lado, se encuentra el software privativo o propietario⁶ que es descrito como antónimo u opuesto de Software Libre y se refiere al software que no es libre, con base en las libertades que serán explicadas más adelante (GNU, 2015). Su principal limitación es restringir el código fuente y condiciones de uso del producto.

Es por ello que dentro del universo de esta clasificación de software, se pueden considerar al software libre y al software privativo son dos conjuntos mutuamente excluyentes. En esta clasificación, el software de fuente abierta tiene características similares que el software libre, se llega a decir que “casi todo el software libre es software de código abierto y casi todo el software de código abierto es software libre”, sin embargo tienen diferencias claras que serán analizadas más adelante (GNU, 2015).

Esta última clasificación servirá como punto de partida para el resto de la investigación, ya que se seleccionaran estos tipos de software, dado sus características, para analizar su proceso de desarrollo. Los tipos de software elegidos son el software libre y el software de fuente abierta, que en términos generales, ambos casos consideran la innovación colaborativa como eje fundamental para el desarrollo de soluciones, a través de la estrategia básica compartir el código fuente y el conjunto de archivos relacionados con el programa.

1.4 Software Libre y de fuente abierta (SLFA)

El software libre y el software de fuente abierta poseen características benéficas para la fomento de capacidades de innovación en el desarrollo de software. La principal característica de estos tipos de software es la posibilidad de compartir el código fuente, o conocimiento codificado, lo permite difundir el conocimiento implícito en la solución y adquirir fácilmente otros códigos fuente, que permiten partir de una base de conocimiento, asimismo estos tipos de software cuentan con la colaboración de muchas personas para desarrollarlos.

Por el contrario, el software privativo restringe el uso, distribución y modificación de los programas, depende de permisos del dueño, desarrollador o distribuidor del software y es complicado obtener el código fuente, se podría obtener mediante contratos de confidencialidad y

⁶ También es conocido en inglés como *non-free software*.

el pago de una alta suma de dinero. De acuerdo a las definiciones de software libre de Free Software Foundation, también se considera como software privativo al software semilibre, que es instalado o usado bajo ciertas condiciones o restricciones del programa, que de alguna forma limita los derechos del usuario sobre el programa (FSF, 2016). Si bien, el código fuente del software era compartido en sus orígenes, en el momento en que se reconoció el valor del software, se prohibió su divulgación sin consentimiento y se restringió los derechos de uso, dando forma a lo que se conoce ahora como software privativo.

Por lo que surgen conceptos como *copyleft* y el movimiento de Software Libre⁷ como alternativa del software privativo, que buscan extender y formalizar la sinergia de colaboración que existe entre algunos grupos de programadores que distribuyen sus programas entre ellos, y además que todo el software sea libre como mecanismo para compartir el conocimiento (Stallman, 2002).

Se considera software libre a todo aquel software del que se tiene la libertad para ejecutar, copiar, distribuir, estudiar, modificar y mejorar el programa, a partir del acceso del código fuente. Las modificaciones y versiones posteriores, derivadas de un software licenciado en este movimiento mantendrán siempre las cualidades de libertad. Este tipo de software está respaldado por la organización fundada por Richard Stallman en 1985, Fundación de Software Libre (FSL)⁸ con una ideología basada en principios morales y éticos (GNU, 2015).

Ilustración 4: Logo de Free Software Foundation



La organización Free Software Foundation (2015) plasma estas cualidades dentro de cuatro libertades esenciales que sus programas deben cumplir. Estas libertades están enumeradas a partir de la libertad cero. La libertad 0 otorga la libertad de uso del programa para cualquier propósito, con el objetivo de consumir la tecnología libremente; la libertad 1 otorga la libertad de estudiar el funcionamiento del programa mediante el código fuente y posibilitando la adaptación y

⁷ El término original, Free Software, es ambiguo ya que <<free>> significa tanto libre como gratuito, sin embargo el concepto de Software Libre no se refiere a lo gratuito, sino a la característica de libertad.

⁸ El nombre original de la organización es Free Software Foundation (FSF)

modificación, lo que promovería capacidades de desarrollo de software y solucionaría problemáticas particulares; la libertad 2 se refiere a la libertad de redistribuir el software que se adquirió (el programa original); mientras que la libertad 3, permite compartir programa que se modificó, permitiendo que el conocimiento generado se haga de dominio público y se beneficie la sociedad (Free Software Foundation, 2015).

Tabla 5: Libertades del Software Libre

Libertades	Descripción
Libertad 0	La libertad de ejecutar el programa como se desea, con cualquier propósito
Libertad 1	La libertad de estudiar cómo funciona el programa, y cambiarlo para satisfacer necesidades particulares. (El acceso al código fuente es una condición necesaria para ello)
Libertad 2	La libertad de redistribuir copias para ayudar a su prójimo
Libertad 3	La libertad de distribuir copias de sus versiones modificadas a terceros. (Esto permite ofrecer a toda la comunidad la oportunidad de beneficiarse de las modificaciones. El acceso al código fuente es una condición necesaria para ello)

Fuente: Elaboración propia basada en Free Software Foundation, 2015.

Los software considerados como Software Libre implican que el programa puede ser usado, copiado y/o distribuido manteniendo las mismas cualidades (libertades), incluso las versiones derivadas. En resumen un Software Libre siempre será libre. Mientras que el Software de Fuente Abierta (SFA) surge una década después, toma de base y antecedente al software libre. Incluye gran parte de las características del software libre, pero añade mayor flexibilidad para hacer negocios.

El software de fuente abierta se caracteriza por tener virtudes pragmáticas y buscar la excelencia técnica, aunque ello implique que los programas derivados dejen de ser libre y tengan características privativas. Por ello el software de fuente abierta coexiste con el software privativo, a diferencia del software libre que lo rechaza tajantemente (OCDE, 2009).

Ilustración 5: Logo de Open Source Initiative



El software de fuente abierta es respaldado por la organización Iniciativa de Fuente Abierta⁹ (IFA), creada por Eric S. Raymond y Bruce Perens en 1995, propone un decálogo de características de las licencias que este tipo de software debe cumplir, basadas en una propuesta directrices del software libre: The Debian (Open Source Initiative, 2015). El decálogo se enlista en la Tabla 4.

Tabla 6: Decálogo de Software de Fuente Abierta

#	Decálogo	Descripción
1	Libre redistribución	El software debe poder ser regalado o vendido libremente
2	Código fuente	El código fuente debe estar incluido u obtenerse libremente
3	Trabajos derivados	La redistribución de modificaciones debe estar permitida
4	Integridad del código fuente del autor	las licencias pueden exigir que las modificaciones sean redistribuidas sólo como parches
5	Sin discriminación de personas o grupos	No se debe restringir el uso a ninguna persona, grupo de personas o región geográfica
6	Sin discriminación de áreas de iniciativa	Los usuarios comerciales no pueden ser excluidos
7	Distribución de la licencia	Deben aplicarse los mismos derechos a todo el que reciba el programa
8	La licencia no debe ser específica de un producto	El programa no puede licenciarse sólo como parte de una distribución mayor
9	La licencia no debe restringir otro software	La licencia no puede obligar a que algún otro software que sea distribuido con el software abierto deba también ser de código abierto
10	La licencia debe ser tecnológicamente neutral	No debe exigirse la aceptación de la licencia por medio de un acceso por clic de ratón o de otra forma específica del medio de soporte del software

Fuente: González, 2015 y Coar, 2015.

El software de Fuente Abierta tiene la libertad de vender o regalar el software como componente de algún otro programa, sin requerir pagos, para facilitar la redistribución; se podrá proporcionar la compilación y el código fuente que permitiría que otros programadores realicen modificaciones al software y se realicen trabajos derivados mediante el mismo licenciamiento que el software original, permitiendo mantener a las distribuciones posteriores con los mismos permisos que la original; contempla la posibilidad de mantener la versión original y solo permitiendo las modificaciones que contribuyan a corregir errores del programa, esto se emplea con la finalidad de mantener la reputación del software y controlar la evolución del software.

⁹ Open Source Initiative (OSI)

Además este software no debe discriminar a ninguna persona o grupo de personas, como lo hacen las restricciones que tiene el software privativo en algunas regiones del planeta; los derechos sobre el programa se transfieren con el programa y no requiere licencias adicionales en cada ocasión que se comparta; todas las distribuciones del programa, dentro de los términos de la licencia, tienen los mismos derechos que la distribución original; el software no puede obligar a los programas que sean distribuidos junto con este a registrarse bajo licencias de fuente abierta; tampoco las licencias pueden restringir a software externo de ninguna forma (González, 2015).

El software libre y el de fuente abierta a pesar de tener en común la mayoría de las características y licencias, tienen claras diferencias que deben ser presentadas para reconocer las particularidades de cada uno de los tipos de software. Las diferencias de estos tipos de software se encuentran descritas en la Tabla siguiente, desde su filosofía, motivación, promotor, organización, año de origen, modelo de negocio, objetivo y la perspectiva sobre el software privativo.

Tabla 7: Comparación de características de software libre y software de fuente abierta

Características	Software libre	Software de fuente abierta
Filosofía	Principios morales y éticos	Virtudes pragmáticas y excelencia técnica
Motivación	Software con libertad de modificación, uso y distribución	Software con mayores ventajas de lucro
Promovido por	Richard Stallman	Eric S. Raymond y Bruce Perens
Organización que lo respalda	Free Software Foundation	Open Source Initiative
Año de fundación	1985	1998
Modelo de negocio	Venta de servicios adicionales al software	Servicios complementarios, licencias mixtas
Producto final	El código fuente siempre es libre	No es necesario proporcionar su código fuente
Opinión sobre el software privativo	El software propietario es antiético por no poderse compartir	El software de fuente abierta es de mejor calidad que el software propietario

Fuente: Elaboración propia basado en Lima Ramos, 2015.

Aunque estos tipos de software no son idénticos, se suele analizar juntos, en un mismo concepto conocido como Software Libre y de Fuente Abierta (SLFA)¹⁰. La virtud de este conjunto de software es la metodología de desarrollo de programas, ya que manejan un esquema de colaboración con usuarios individuales y comunidades de desarrollo. Lo que implica expandir el

¹⁰ Conocido en inglés como Free and Open Source Software, FOSS o FLOSS

potencial del departamento de Investigación y Desarrollo (I+D) más allá de las limitaciones de la organización o empresa (Oppenheimer, 2014).

1.5 Propiedad Intelectual y licencias de software

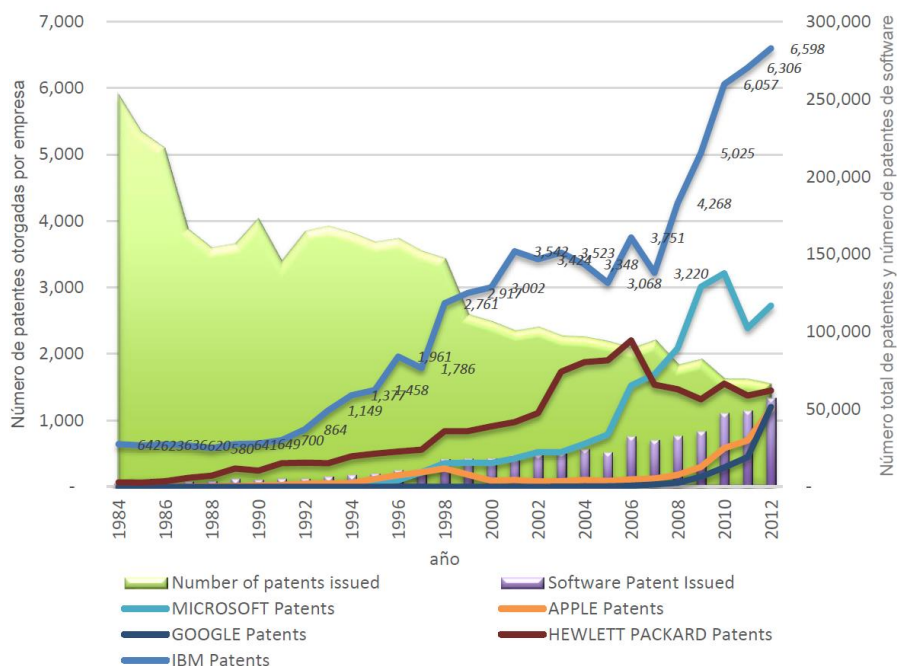
Aunque el tamaño e importancia de la industria del software son considerables, su adecuada protección del software sigue siendo un reto. El software suele protegerse mediante la figura de derechos de autor, ya que el software se plasma mediante lenguaje de computadora que puede interpretarse como una obra literaria, su protección está vigente en el Tratado de la Organización Mundial de la Propiedad Intelectual (OMPI) sobre Derechos de Autor (OMPI, 2008).

Sin embargo la protección del software bajo Derechos de Autor es débil ya que no contempla los aspectos funcionales del software y considera al código fuente de una computadora de manera similar una expresión literaria, aunque se proteja el código fuente, no se protegen sus funciones técnicas, por lo que pueden ser replicadas empleando un código diferente o incluso otro lenguaje de programación.

Algunas Oficinas de Propiedad Intelectual como la de Estados Unidos han realizado esfuerzos, desde inicios de la década de 1980, para proteger el software mediante una figura de Propiedad Intelectual más fuerte, la patente. Los países que consideran la protección del software por patente son Estados Unidos, Australia, Brasil, India y Japón (OMPI, 2008). En otros países que no contemplan la protección del software por patente, emplean una estrategia de protección indirecta, conocida como Software implementado por computadora.

La oficina de patentes de Estados Unidos, United States Patent and Trademark Office (USPTO), tiene una tasa de patentes concedidas que se ha ido reduciendo a través del tiempo, sin embargo las patentes de algunas de las principales empresas desarrolladoras de software tienen un crecimiento de patentes concedidas como se muestra en la siguiente Gráfica.

Gráfica 6: Número de patentes otorgadas por la USPTO

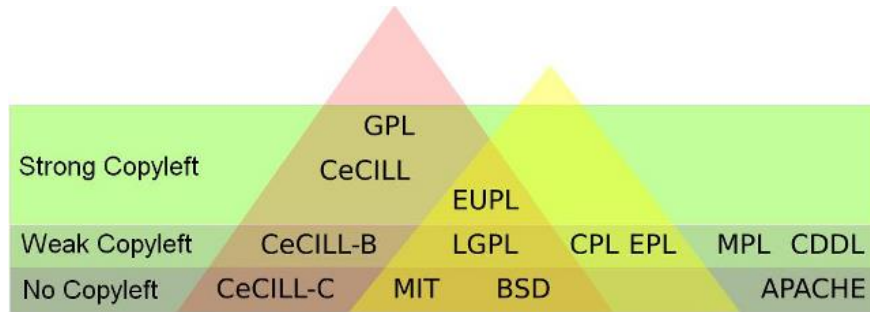


Fuente: Borja, 2015.

A pesar de los esfuerzos, aun no hay tratados internacionales que se refieran a la protección del software como patente. Por lo que las licencias son es esquema funcional más parecido a esta protección, las cuales son un contrato entre el usuario consumidor y el titular del software, descritas a través de una serie de términos y condiciones.

Las licencias del Software Libre y de Fuente Abierta son numerosas, algunas de ellas son las licencias propuestas Free Software Foundation y por Open Source Initiative, además otras organizaciones públicas y privadas proponen modelos propios de licencia, como Eclipse, Microsoft, Mozilla, Nokia, Python, la Unión Europea, la NASA, entre otros que están enlistados en el Anexo 2 (CENATIC, 2009). Cada licencia otorga ciertos derechos o permisos al usuario, por lo que estas licencias se pueden clasificar como copyleft fuerte, copyleft débil y sin copyleft.

Ilustración 6: Compatibilidad de licencias



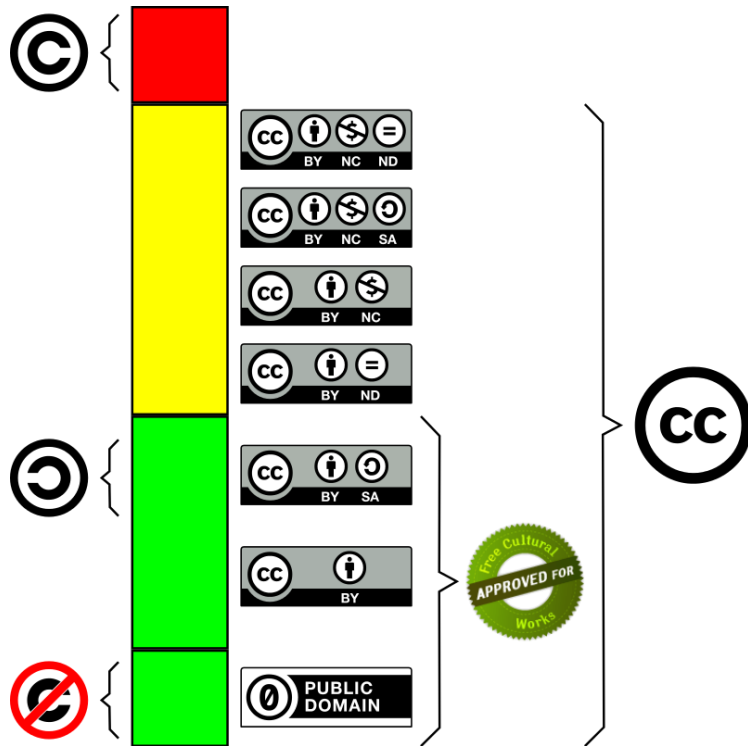
Fuente: GNU, 2015.

Las licencias condicionan a los usuarios sobre las posibilidades de uso, copiado, modificación y redistribución. Por un lado, el software que no tiene licencia es considerado de dominio público, puede ser usado, copiado, modificado y redistribuido libremente, sin embargo no tiene garantías de que mantenga las mismas libertades. Mientras que por el otro lado, hay software que no es distribuido y no requiere licencia, así que se maneja como secreto comercial, como lo es en el caso del software embebido.

Aunque también existe un grupo de licencias llamadas Creative Commons (CC), muy comunes no solo en el software, sino también en fotos, música, obras literarias y otros. Las licencias de Creative Commons buscan regular la protección de bienes inmateriales, se rigen bajo contratos de licenciamiento y buscan ayudar a compartir legalmente el conocimiento. La organización fue fundada en 2001 con ayuda del Center for the Public Domain y su junta directiva se compone por líderes de opinión, expertos en educación, juristas, inversionistas, empresarios y filántropos (Vercelli, 2004).

En el 2002, se publicó el primer conjunto de licencias Creative Commons de derechos de autor inspiradas parcialmente por las licencias General Public License de la Fundación de Software Libre (GNU GPL) (Creative Commons, s.f.). El grupo de licencias de Creative Commons se pueden explicar mediante su nivel de restricción de derechos, como un semáforo, en donde se colocan en rojo las licencias convencionales que son muy restrictivas, en amarillo algunas licencias CC que permiten copiar, modificar y publicar software y otras obras bajo determinadas circunstancias, mientras que las licencias en verde son aquellas que no ponen restricciones (Ikusimakusi, 2012)

Ilustración 7: Semáforo de licencias Creative Commons



Fuente: Ikusimakusi, 2012.

CAPITULO 2: PROCESO DE INNOVACIÓN DEL SOFTWARE LIBRE Y DE FUENTE ABIERTA

2.1 Conceptos de Innovación y conocimiento

Diversos estudios, desde las aportaciones de Solow hasta algunas más recientes como las de Romer y Lucas, demuestran el impacto positivo que tienen los cambios tecnológicos e innovaciones en el desarrollo económico (Aboites, 2007), creando mercados más competitivos que benefician a la sociedad en diversos contextos como en lo social, cultural, político, moral, etcétera. (Echeverría, 2013). Shumpeter describía este fenómeno en su concepto de Destrucción creativa, que se refiere al proceso de reemplazo de tecnología previa por innovaciones, lo que reconfigura el mercado y plantea nuevos escenarios, ya que son las innovaciones las que estimulan la reestructuración, reinención y revolución del mercado (OCDE, 2003).

Por ello, es necesario partir de definiciones concretas para construir las bases de ésta investigación, se debe diferenciar innovación, invenciones e ideas. En primer lugar, el concepto de ideas se refiere a las representaciones mentales de algo, independientes del mundo físico, es decir parten del razonamiento o la imaginación de las personas sin llegar a materializarse de ningún modo. Por otro lado, las invenciones son todas aquellas creaciones humanas nuevas que satisfacen sus necesidades concretas mediante la transformación de la materia o la energía que existe en la naturaleza (LPI, 1994).

Mientras que el concepto de innovación podría partir de una definición básica como la creación o modificación de un producto que es llevado al mercado (Real Academia de la Lengua Española, s.f), sin embargo la definición de este concepto es limitada, por lo que para esta investigación se emplearán otras definiciones como las del Manual de Oslo de la OECD.

“Una innovación es la introducción de un nuevo, o significativamente mejorado, producto (bien o servicio), de un proceso, de un nuevo método de comercialización o de un nuevo método organizativo, en las prácticas internas de la empresa, la organización del lugar de trabajo o relaciones exteriores.” (OECD, 2005)

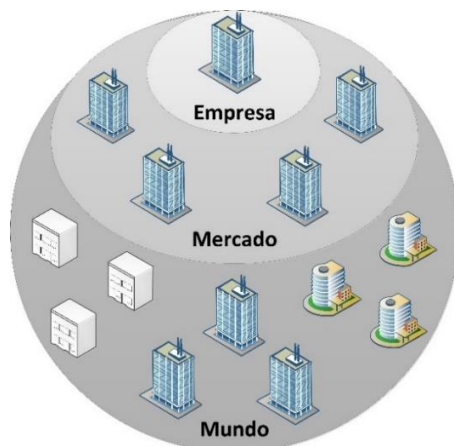
Esta definición permite establecer un marco de referencia sobre el concepto de innovación y sus principales tipos, como: innovación de producto, proceso, mercadotecnia y organización. En el caso del software, a pesar de ser un elemento intangible tiene características técnicas que satisfacen necesidades particulares por lo que también podría ser considerado como

invención e innovación, particularmente el concepto de innovación tecnológica, que incluye los dos primeros tipos, la innovación de producto y la innovación de proceso:

- Innovación de producto: enfocada en productos o servicios nuevos o significativamente mejorados, en cuanto a sus características o uso. Incluye la mejora significativa de sus componentes, materiales y características técnicas, entre otros factores.
- Innovación de proceso: considera a los procesos de producción o de distribución, que implica cambios significativos en las técnicas, materiales y/o programas informáticos. Su objetivo es reducir los costes de producción o distribución, mejorar la calidad, entre otros factores.

En este sentido, es necesario definir también el concepto de novedad, ya que las innovaciones tecnológicas las describimos como la introducción de un nuevo o significativamente mejorado producto, servicio, proceso de producción o proceso de distribución. Se dice que algo es nuevo si no está incluido en el estado del arte o estado de la técnica de los productos, servicios o procesos, por ello es necesario considerar el grado de novedad de las innovaciones de acuerdo a una referencia espacial, ya sea nuevo para la empresa, nuevo para el mercado o nuevo para el mundo (OCDE, 2005). Se muestra gráficamente la explicación de grado de novedad con los tamaños de los círculos de la siguiente ilustración.

Ilustración 8: Grado de novedad de las innovaciones



Fuente: Elaboración propia basada en el Manual de Oslo, 2005.

La clasificación del grado de novedad depende del tamaño del entorno en el que se considera a algo como nuevo. Nuevo para la empresa tiene un mínimo nivel de exigencia, ya sea

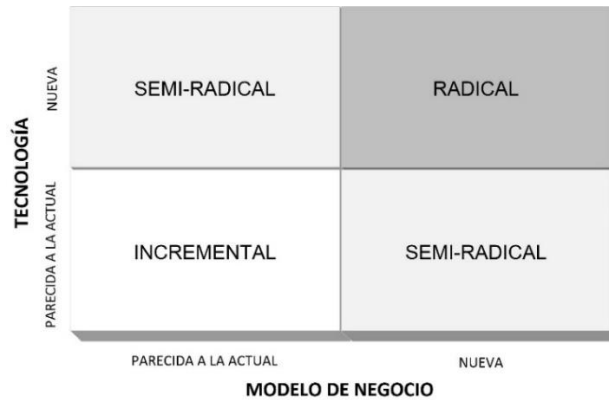
que la empresa adopta o desarrolla algo que no existe dentro de ésta; nuevo para el mercado tiene un nivel de exigencia intermedio, ya que se considera nuevo dentro las empresas del sector que se consideran competidores; y nuevo para el mundo entero se considera cuando la empresa es la primera en lanzarlo en todos los mercados y sectores del mundo (OCDE, 2005).

Si bien, el grado de la novedad es un factor importante, no puede correlacionarse directamente con el impacto de las innovaciones, ya que ser una invención nueva para el mundo no implica que tenga un mayor impacto que los otros grados de novedad. En el proceso de desarrollo de Software Libre y de Fuente Abierta, se parte de software/ innovaciones nuevas para la empresa, que en conjunto con las capacidades de tecnológicas de la organización, para crear principalmente productos/ innovaciones nuevas para el mundo, ya que el mercado del software es básicamente global.

En el proceso de innovación, las nuevas tecnologías reemplazan a las antiguas de dos formas, mediante una innovación radical que produce grandes cambios en el mundo o a través de innovaciones progresivas que produce pequeñas mejoras incrementales. Los procesos de innovación dependen de cada sector, por ejemplo en el sector de las TIC y el software, los productos tienen ciclos de vida muy cortos, por lo que los cambios son rápidos y algunos de estos cambios implican una ruptura en el mercado, por su impacto (OCDE, 2005).

Las innovaciones radicales implican una ruptura en la tecnología actual, tienen un impacto significativo en el mercado, ya sea cambiando su estructura o creando nuevos mercados. Por el contrario, las innovaciones incrementales se basan mejoras progresivas a la tecnología existente, en el mismo mercado. Y por otro lado, las innovaciones semi-radicales son mejoras a tecnologías parecidas a las conocidas que se introducen a un nuevo mercado o si se encuentra en el mismo mercado, actualiza la tecnología implicada (Dávila, Epstein, & Shelton, 2006). Estos conceptos son descritos gráficamente en la siguiente ilustración.

Ilustración 9: Matriz de la innovación



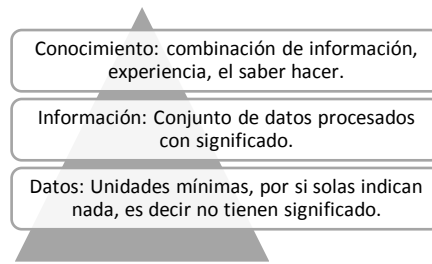
Fuente: Elaboración propia a partir de Dávila, Epstein, y Shelton, 2006.

Por otro lado, también es necesario reconocer los conceptos vinculados al conocimiento y la relación que tiene con la innovación. En el proceso de desarrollo de software se manejan y transfieren tecnología en tres principales dimensiones: datos, información y conocimiento (Sáez Vacas, 1991).

- Datos: Son observaciones simples de algunos sucesos, por lo que se adquieren, se organizan, se cuantifican y se transfieren con facilidad.
- Información: Son datos que tienen un propósito y pertinencia, por lo que se requiere una unidad de análisis, un consenso sobre significados y la interpretación humana.
- Conocimiento: Es información valiosa reconocida por la mente humana, por lo que es difícil de estructurar, capturar y transferir.

Siendo este último, el concepto que mejor describiría al software, ya que el software es considerado como conocimiento codificado que implica cierto nivel de complejidad y abstracción, aunque su codificación es parte del lenguaje común de un conjunto de personas con conocimientos técnicos y especialistas en software, tales como programadores, ingenieros de cómputo, administradores de redes, entre otros más.

Ilustración 10: Datos, información y conocimiento



Fuente: Elaboración propia.

La comprensión y generación de software requiere de conocimiento previo y a la capacidad cognitiva de las personas, es decir, entre mayor sea el conocimiento del receptor requiere menor esfuerzo para procesar determinada información y generar nuevo conocimiento. Por ello, las personas son el factor clave del proceso de gestión del conocimiento, interpretando y procesando los símbolos que generan nuevos significados y procesos, plasmados principalmente a través del código fuente (Sáez Vacas, 1991).

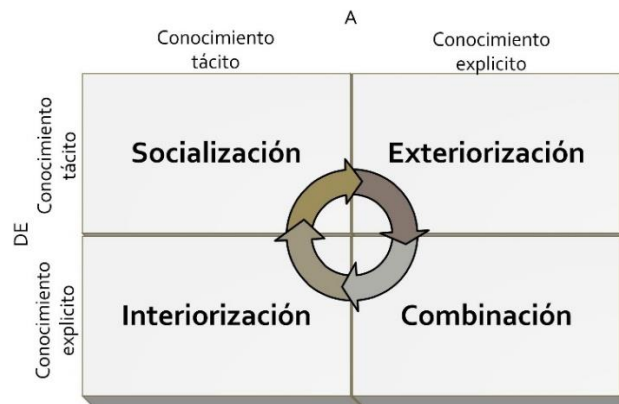
La gestión del conocimiento se basa en un conjunto de procesos sistemáticos que identifican, captan y procesan conocimiento externo, para mejorar ventajas competitivas en las organizaciones o individuos. La gestión adecuada del conocimiento codificado de las organizaciones que son parte del ecosistema de desarrollo de software les permite compartir y reconocer buenas prácticas, desarrollar inteligencia competitiva, reducir el tiempo de resolución de problemas, disminuir costos y mejorar la eficiencia global de las organizaciones (Rodríguez Gómez, 2006).

La gestión del conocimiento es un proceso que depende del conocimiento previo, de la capacidad de absorción de conocimiento externo, sin embargo de acuerdo a Nonaka y Takeuchi (1995, 1999), la gestión del conocimiento implica dos dimensiones: la dimensión ontológica del conocimiento y la dimensión epistemológica de conocimiento.

La dimensión ontológica del conocimiento reconoce que la creación de conocimiento organizacional depende de la creatividad individual que en conjunto crean una red de conocimientos de la organización, por lo que las fuentes de conocimiento son los individuos y las agrupaciones que se consoliden. Mientras que la dimensión epistemológica se refiere al proceso de comunicación y la interacción o conversión del conocimiento tácito y del conocimiento explícito.

- Conocimiento tácito: Este conocimiento es personal, compuesto de las ideas, habilidades y valores de cada individuo. Es difícil de comunicar y transferir.
- Conocimiento explícito: Se expresa de manera formal y se puede empaquetar. La transferencia de conocimiento es posible, si el receptor posee las claves y conocimientos para aprovecharlo.

Ilustración 11: Proceso de conversión de tecnología



Fuente: Nonaka y Takeuchi, 1995.

La conversión o transferencia de conocimiento tiene características particulares que son descritas a continuación:

- **Socialización.** De conocimiento tácito a conocimiento tácito: Mediante una interacción de personas, se transmiten los conocimientos directamente.
- **Externalización.** De conocimiento tácito a conocimiento explícito: Para esta transferencia de tecnología se requiere que el conocimiento de una persona se plasme mediante esquemas, fórmulas y métodos.
- **Combinación.** De conocimiento explícito a conocimiento explícito: Los manuales, esquemas o datos se reinterpretan, se mezclan para crear nuevo conocimiento explícito.
- **Internalización.** De conocimiento explícito a conocimiento tácito: Las personas asimilan conocimiento plasmado en documentos y lo convierte en parte de sus conocimientos y experiencias.

2.2 Modelos de gestión de la innovación

El proceso de desarrollo de un nuevo producto o invención suele considerarse como resultado de un proceso creativo, en el que se emplea los recursos disponibles tanto económicos como intelectuales para materializar una idea en una invención que resuelva algún problema técnico. Una invención requiere introducirse en el mercado para que sea considerada como innovación, por lo que en este proceso se requiere de la capacidad de comercialización de un equipo de ventas.

Es decir una empresa, organización o individuo requiere gestionar sus capacidades tecnológicas y sus capacidades de comercialización para conseguir un eficiente proceso de innovación. Debido a la necesidad de manejar eficientemente los recursos de las organizaciones para mejorar el número y éxito de las innovaciones, han surgido modelos de gestión de la innovación que son guías para innovar de manera metódica y sistematizada.

El paradigma del proceso innovación se ha transformado, desde los modelos cerrados y lineales, en los que las empresas desarrollan sus nuevos productos a su interior, en sus departamentos de I+D. Las empresas bajo estos modelos buscan contratar a gente inteligente, impulsar por si mismas la Investigación y Desarrollo (I+D), introducir primero sus innovaciones y controlar la propiedad intelectual de sus desarrollos. Mientras que los paradigmas más recientes, consideran las constantes interacciones que tienen las empresas con agentes externos, tanto en las entradas como en las salidas del proceso de innovación (Chesbrough, 2003).

Cada modelo de gestión de la tecnología representa una perspectiva sobre el proceso de innovación. Es por ello, Rothwell en 1992, distinguió cinco generaciones de modelos del proceso de innovación, mediante un análisis de los modelos conocidos. Aunque aclara que la secuencia de las generaciones no implica una jerarquía sobre cuál es el mejor modelo, que todos los modelos de gestión de la innovación son eficientes, solo depende de cuál de ellos se adapte mejor a cada una de las empresas (Žižlavsk, 2013).

- Primera generación (1950s- Mitad de 1960s): Modelo de innovación empujado por la tecnología.

Este modelo lineal está constituido de fases cronológicas, desde la investigación básica, la de diseño e ingeniería, producción, mercadotecnia y ventas. En esta última es la salida de un nuevo producto exitoso. Surge de la investigación de científicos y académicos, que en la mayoría de los casos no ven aplicación práctica de sus investigaciones hasta en una etapa avanzada. El láser es ejemplo de este modelo, ya que primero se realizó investigaciones para desarrollar modelos teóricos, para posteriormente construir prototipos y actualmente siguen surgiendo usos en muchos campos como en el médico o en las tecnologías de la información.

Ilustración 12: Modelo lineal de innovación empujado por la tecnología



Fuente: Elaboración propia basada en Rothwell, 1994

- Segunda generación (Mitad de 1960s- Principios de 1970s): Modelo de innovación atraída por la demanda.

Utilizada principalmente en la década de 1960s a la primera mitad de la década de 1970s. Debido al incremento de competencia y diversificación de productos/servicios, se volvió importante considerar las necesidades de los clientes tanto en el proceso de innovación como en la mercadotecnia. Es decir, las demandas del mercado son las que determinan el rumbo de los departamentos de Investigación y Desarrollo de las empresas para generar productos que sean requeridos por los consumidores.

Ilustración 13: Modelo lineal de innovación atraída por la demanda

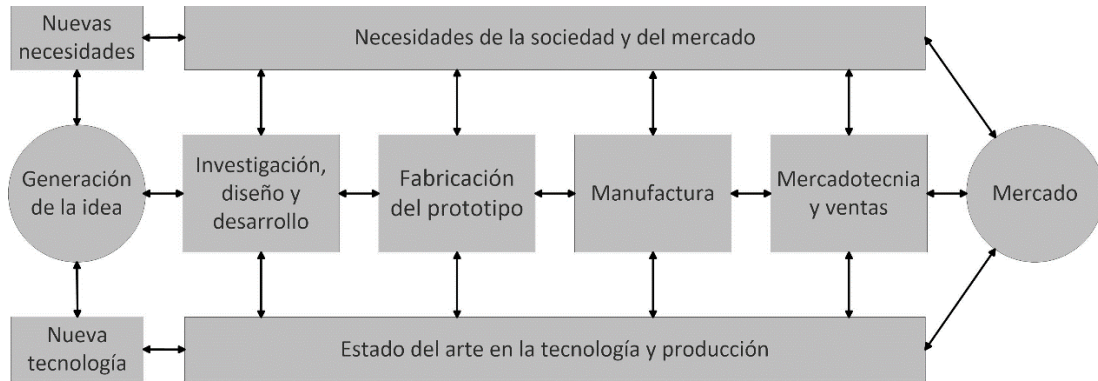


Fuente: Elaboración propia basada en Rothwell, 1994

- Tercera generación (Principios de 1970s- Mitad 1980s): Unión de la I + D y comercialización (el modelo interactivo).

Fue resultado de la crisis económica posterior a la Segunda Guerra Mundial, por las altas tasas de inflación y saturación de la demanda, así como del crecimiento del desempleo, lo que provocó que las compañías adoptaran estrategias de racionalización y consolidación. Ya que ni el empuje de la tecnología, ni la atracción del mercado fueron suficientes, por lo que se tuvieron que unir ambos modelos, agregar fases e incluir retroalimentación.

Ilustración 14: Modelo de innovación mezclado

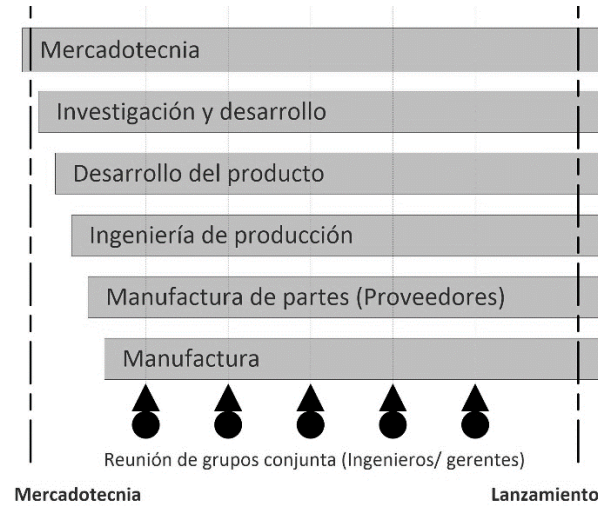


Fuente: Elaboración propia basada en Rothwell, 1994

- Cuarta generación (Principios de 1980s-Principios de 1990s): Modelo integrado o encadenado.

Se caracteriza por el uso paralelo de los equipos de investigación integrados, además de involucrar al proveedor y clientes importantes. Inicialmente las empresas concentraban las competencias centrales tanto de la tecnología como del negocio, mientras que en esta generación existe una colaboración horizontal de investigación, desarrollo y producción, incluso al exterior de la empresa. Este modelo reconoce la necesidad de integrar el conocimiento en el proceso de innovación, aunque no precisamente conocimiento científico, sino el que se genera de la interacción entre individuos de la compañía, la compañía misma y su entorno (Žižlavsk, 2013).

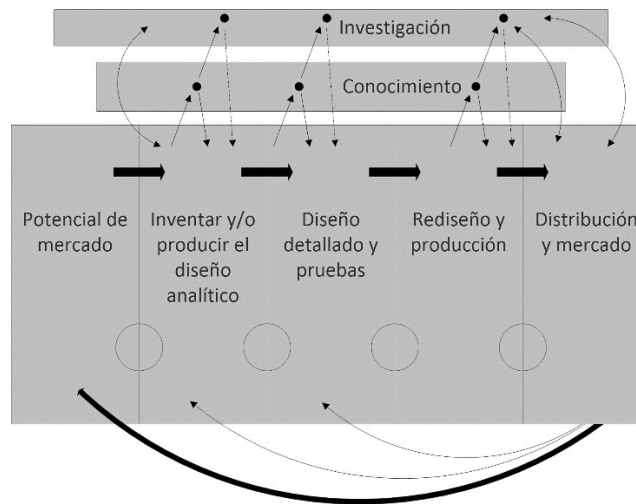
Ilustración 15: Modelo de innovación integrada con funciones paralelas



Fuente: Elaboración propia basada en Rothwell, 1994

Debido al crecimiento de la competencia, a la reducción del ciclo de vida del producto y al cambio tecnológico, se incluyó una estrategia basada en el tiempo, lo que significó cierta ventaja competitiva

Ilustración 16: Modelo de innovación integrada con retroalimentación



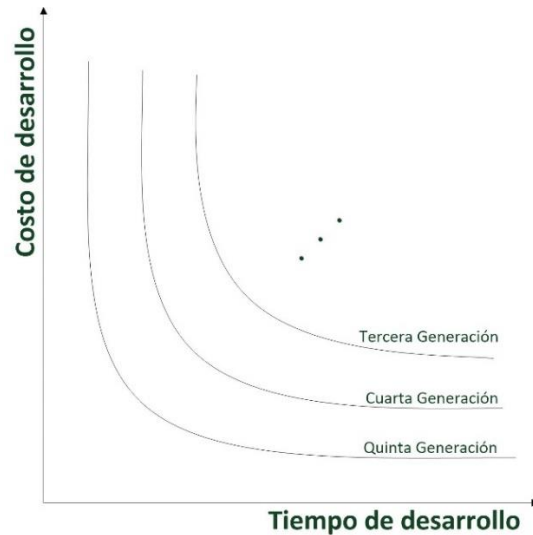
Fuente: Elaboración propia basada en Žižlavsk, 2013.

- Quinta generación (Principios de 1990s): Modelo de sistemas y redes.

Las nuevas condiciones del mercado han forzado a las empresas a adoptar nuevos métodos para mejorar la eficiencia del proceso de innovación. Las medidas buscan mejorar la relación de

costos respecto a las generaciones de gestión de la innovación anteriores, ya que los primeros modelos permitían gestionar los recursos de manera sistemática para innovar.

Ilustración 17: Curvas de costos y tiempo de desarrollo



Fuente: Elaboración propia basada en Žižlavsk, 2013

Por lo que, cada uno de los modelos subsecuentes ha tratado de tener una mejor relación de los costos de investigación y el tiempo de desarrollo, con ello se lograría pasar a una curva de costos más eficiente en el transcurso de las generaciones de modelos de gestión de la innovación. Si se busca continuar aumentando la eficiencia se requiere: i) un sistema de integración con la organización interna, ii) extender las redes, iii) estructuras organizacionales flexibles y planas, iv) prohibir la madures de los datos internos y v) el desarrollo de productos con soporte electrónico.

Esta quinta generación representa una transición a los medios electrónicos, empresas avanzadas que utilizan métodos de TI, para apoyar y acelerar el proceso de innovación. Siendo el internet el mejor impulsor de expansión de I+D y facilitando la integración con factores del entorno de la empresa como competidores, comerciantes, clientes, proveedores, etcétera.

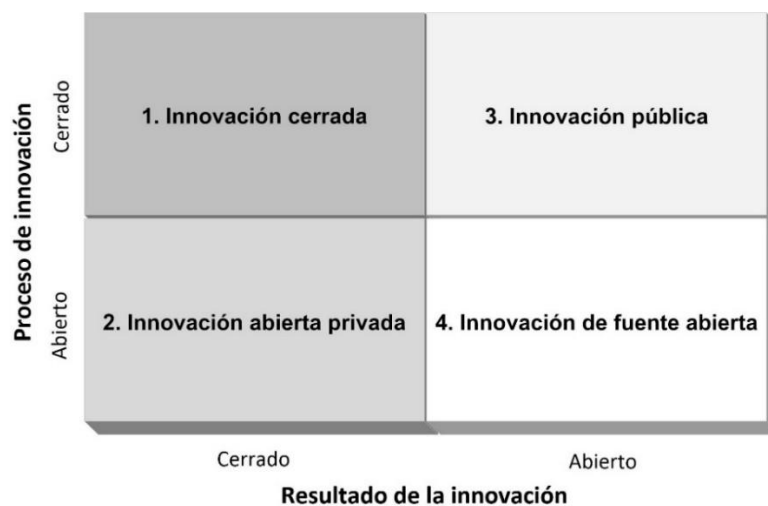
Los cambios económicos y sociales como la tercerización del trabajo, la colaboración laboral a distancia mediante las nuevas tecnologías y la apertura de nuevos mercados, han provocado la adopción de esquemas de negocio y de gestión de tecnología alternativos que se adapten de mejor forma a las nuevas condiciones del mercado. Por ello, modelos como el de innovación abierta han

tenido mayor aceptación en las empresas, principalmente a las del sector de Tecnologías de la información y comunicación (Huizingh, 2011).

Esta quinta generación de modelos de gestión de la innovación se caracteriza por la integración de sistemas y redes, (Žižlavsk, 2013) tal como el modelo de Innovación abierta que plantea Henry Chesbrough en el 2003. Es por ello que se elegirá el modelo de innovación abierta como marco metodológico de análisis debido a que describe de mejor manera el proceso de desarrollo de software libre y de fuente abierta.

Chesbrough (2003) definía a la innovación abierta como el uso de intencional de los flujos de conocimiento de entrada y salida para acelerar la innovación interna, y ampliar los mercados para uso externo de la innovación. Es decir, el modelo conecta el proceso de adquisición de conocimiento externo y el proceso de explotación del conocimiento interno de manera sistemática. Posteriormente Huizingh (2011) propone una clasificación mediante una matriz en la que analiza las prácticas de la innovación en cuanto al proceso de innovación y en su resultado con lo que se determinaría que enfoque de la estrategia que se está buscando al emplear el modelo de innovación abierta.

Ilustración 18: Matriz de condiciones del proceso y resultado de la innovación



Fuente: Elaboración propia basada en Huizingh, 2011

La estrategia es de innovación cerrada si las condiciones tanto en el proceso como en el resultado de la innovación son cerrados, es por ello que la I+D, la fabricación y el lanzamiento son

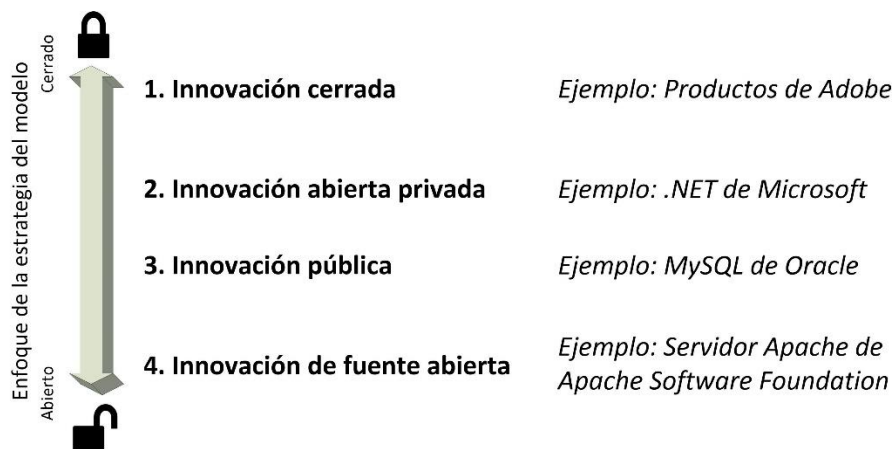
procesos realizados al interior de la empresa, buscando que la empresa sea la única que conozca sobre el conocimiento implícito en la innovación, tal como sucede con los programas de Adobe.

La estrategia es de innovación abierta privada si el proceso de innovación es abierto y se restringen los resultados de la innovación, como en el caso de las empresas que tienen un buen sistema absorción del conocimiento y que permiten la colaboración de desarrolladores externos, pero las empresas guían el objetivo del proyecto, tal como Microsoft lo hizo al liberar el código fuente de .NET en el repositorio GitHub, bajo la licencia “MIT License” (Quijano, 2014).

La estrategia es de innovación pública si el proceso de innovación es cerrado, pero el resultado es abierto. Tal como MySQL de Oracle Corporation, una de las bases de datos de fuente abierta más populares, que tiene varias versiones, una con licencia de software libre y otras con licencias de fuente abierta que permiten incorporar el software el software privativos mediante un pago.

La estrategia es de innovación de fuente abierta si tanto el proceso de desarrollo como el resultado son abiertos, es decir que se desarrolla la innovación a partir del conocimiento público y se divulga y explota libremente, sin fines de lucro. Ejemplo directo del software libre como Apache o GNU/Linux.

Ilustración 19: Enfoque de estrategias del modelo de gestión de la innovación



Fuente: Elaboración propia.

En este sentido, se tiene una gama enfoques de estrategias de innovación, desde una innovación cerrada en el que todo el proceso de innovación se desarrolla al interior de la empresa,

similar a lo que sucede con el software privativo, hasta un enfoque de estrategia de innovación de fuente abierta en el que tanto el proceso como el resultado de la innovación es abierto, similar al software libre. Los enfoques de estrategias intermedias como la innovación abierta privada o innovación pública son alternativas híbridas de los dos enfoques anteriores, pueden servir como estrategias para políticas innovación en empresas o de políticas públicas de gobierno.

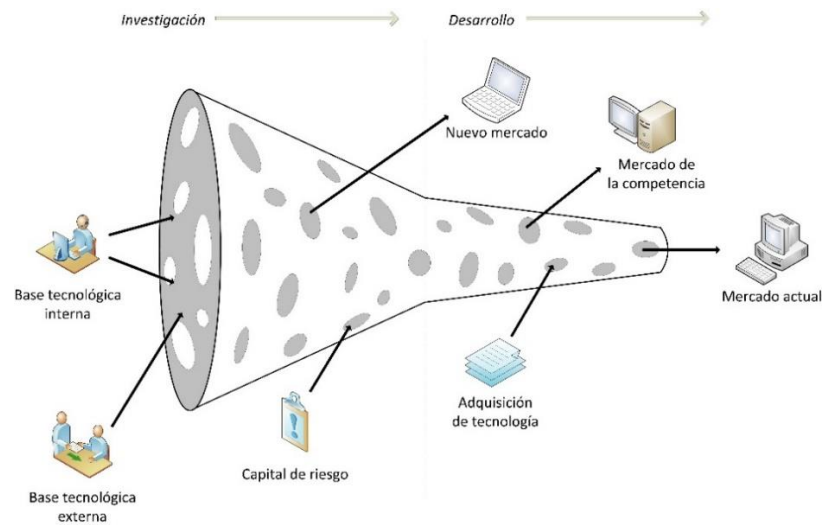
Por ello, la innovación abierta es un modelo que busca vincularse con personas inteligentes dentro y fuera de la empresa, absorber conocimiento generado por I+D externa, construir un mejor modelo de negocios, reconocer las mejores ideas del sector y encontrar esquemas de beneficios al compartir la propiedad intelectual interna (Chesbrough, 2003).

El software libre y de fuente abierta son tipos de software que emplean esquemas de colaboración abierta, en la que se divide el proyecto en pequeñas partes y numerosas personas aportan su trabajo, de esta forma el tamaño de los proyectos y la velocidad de desarrollo que se pueden realizar es de mayor escala. En algunos casos, se plantea que la tecnología desarrollada es un bien común, al ser una producción social y puede utilizarse o modificarse libremente (Benkler, 2005).

El modelo de innovación abierta es un sistema que explora las oportunidades de innovación de fuentes internas y externas a la empresa, las integra con sus capacidades y recursos, para explotarlas por múltiples canales. Por lo que este modelo explica adecuadamente la innovación colaborativa y los resultados, que son la base de estos tipos de software que desarrollan colaborativamente y en muchos casos los derechos para usar la tecnología son libres (West, 2006).

El modelo de innovación abierta tiene dos fases en el proceso de innovación: la de investigación y la de desarrollo. La primera fase inicia con la base tecnológica de la empresa de proyectos de investigación interna, pero se fortalece de otros proyectos de investigación externos a la empresa, así como de capital de riesgo. Por otro lado, si es necesario se adquiere tecnología externa que complementa el proyecto que se está desarrollando.

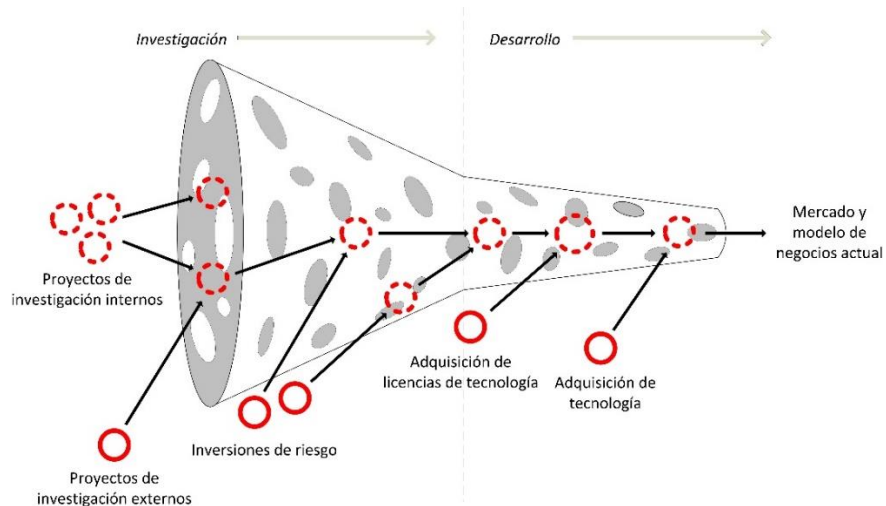
Ilustración 20: Modelo de innovación abierta



Fuente: Elaboración propia basada en Chesbrough, 2003.

En otras palabras, la empresa emplea insumos propios como parte de una base tecnológica, sus capacidades tecnológicas y recursos internos, pero el modelo está abierto a fortalecerse o complementarse con recursos y capacidades externas, para enfocarse a resolver una problemática particular en el mercado que se desenvuelve. Se sigue complementando de conocimiento tecnológico externo durante el proceso, llamado innovación abierta de entrada.

Ilustración 21: Innovación abierta de entrada

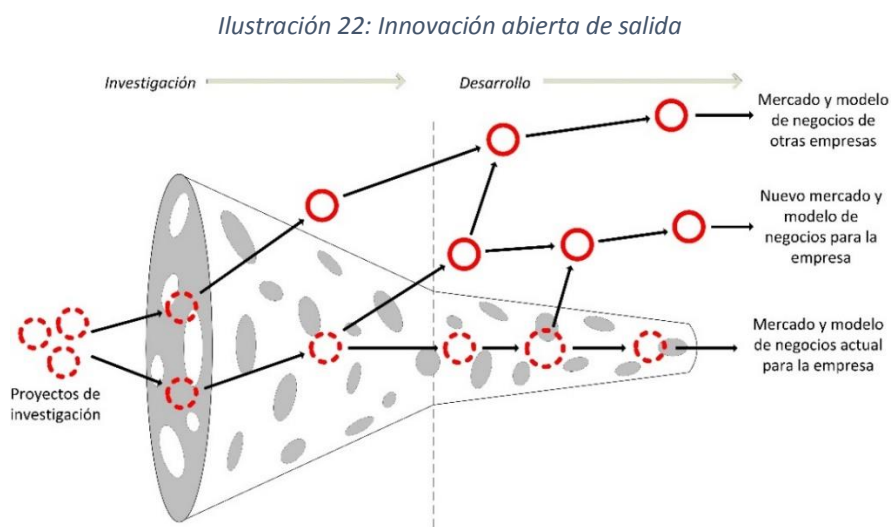


Fuente: Elaboración propia, basada en Chesbrough, 2003.

Por otro lado, en la innovación abierta de salida se considera inicialmente los objetivos que se desean conseguir en el proceso de desarrollo del proyecto, pero con una vigilancia permanente

en los posibles usos que tenga el conocimiento que se está generando en la empresa. El modelo contempla la posibilidad de que la tecnología pueda aplicarse en otro mercado o incluso que la innovación cree un nuevo mercado y modelo de negocio.

Las salidas del modelo de innovación abierta es conocimiento tecnológico interno que tiene aplicación fuera de la empresa con nuevos productos en el mismo mercado o en diferentes mercados, ya sea que la misma empresa explote la innovación o la venda a terceros mediante licencias o spin off.



Fuente: Elaboración propia basada en Chesbrough, 2003.

2.3 Innovación colaborativa y gestión de la innovación en el software libre y de fuente abierta

Las estructuras de las empresas y organizaciones deben evolucionar para acelerar y mejorar la eficiencia de la innovación, pero sobre todo para adaptarse a los cambios tecnológicos y de mercado, aprovechando oportunamente las nuevas herramientas o desarrollando otras que le otorguen ventajas competitivas. La finalidad de aprovechar las nuevas tecnologías consiste en resolver los retos de la sociedad de forma sostenible, rentable y más eficazmente que antes (Curley, 2016).

De igual manera los modelos de gestión de la innovación han evolucionado para responder mejor a los cambios del entorno y reconocer las ventajas de los modelos de Investigación y desarrollo abiertos, que empleen un proceso de innovación colaborativa. Algunos autores como Curley, Grove, introducen el término de innovación abierta 2.0 para referirse a la última evolución del modelo de innovación abierta en el que se propone una mayor integración con agentes externos que colaboran en el proceso de innovación.

Ilustración 23: Evolución del modelo de innovación abierta



Fuente: Elaboración propia.

La innovación cerrada busca a gente talentosa para que trabaje en la empresa, desarrolla las nuevas ideas para crear invenciones que introduce en el mercado depende de una serie de condiciones para que la empresa pueda desarrollar innovaciones; por su lado la innovación abierta reconoce que no toda la gente talentosa trabaja en la empresa, pero crea redes de colaboración para aprovechar el talento interno y externo; mientras que la innovación abierta 2.0 parte de los principios de la innovación abierta pero con un enfoque más abierto e integrado con agentes externos como proveedores, clientes y competidores.

Tabla 8: Comparativo de modelos de gestión de la innovación

Característica	Innovación cerrada	Innovación abierta	Innovación abierta 2.0
Relación con otros	Dependencia	Independencia	Interdependencia
Proveeduría	Subcontratación	Concesión de licencias	Fertilización cruzada cruzadas
Vinculación	Aislado	Bilateral	Ecosistema
Gestión del proceso	Control	Administración	Orquestación
Negociación	Ganar-perder	Ganar-ganar	Ganar más-ganar más
Redes	Subcontratos lineales	Bilateral	Triple o cuádruple hélice

Fuente: Elaboración propia a partir de Curley, 2016.

Aunque se manejen ciertas diferencias entre los modelos de innovación abierta e innovación abierta 2.0, el desarrollo colaborativo es la base de ambos. Su importancia se observa con los siguientes movimientos como Open source, Maker y Crowdsourcing, conceptos como economía solidaria, wkinomy y shareconomy o incluso políticas como capital social y valor compartido. En estos ejemplos recientes sobre sale el proceso de innovación colaborativa o colectiva.

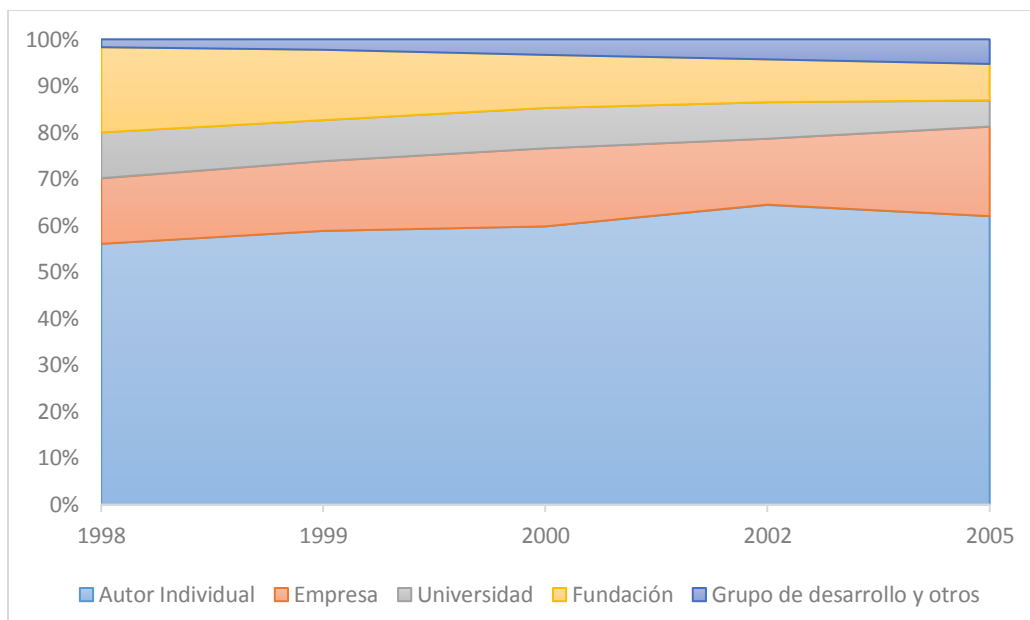
El modelo de innovación abierta tiene el mismo comportamiento que el proceso de desarrollo de software libre y de fuente abierta, mientras uno busca compartir y mejorar las soluciones, el otro busca lo mismo para el código fuente, y ambos cuentan con un ecosistema a su alrededor con el que interactúan e intercambian conocimiento. Es decir, tanto el software libre y de fuente abierta como el modelo de innovación abierta, tienen son componentes clave: primero se comparten los derechos para usar la tecnología y segundo se desarrolla colaborativamente la tecnología usando la aportación de trabajo (West & Gallagher, 2006).

2.4 Agentes involucrados en el proceso de desarrollo del software libre y de Fuente abierta

Los desarrolladores de Software libre y de Fuente Abierta (SLFA) aparecieron con el nacimiento del software, de manera informal, compartiendo sus códigos y aplicaciones de persona a persona. Sin embargo es hasta los esfuerzos de Richard Stallman con su fundación y la aportación de código (GNU) es que el movimiento toma forma y se plantean las primeras licencias que posibiliten el trabajo colaborativo y la libertad del software. Resultado de ello, años más tarde, surge Linux (también conocido como GNU-Linux), basado en el sistema operativo UNIX y aparecen las primeras comunidades de desarrollo como Apache y Debian (Sampedro, 2012).

Los principales desarrolladores de SLFA han sido usuarios individuales y comunidades de desarrollo, aunque en los últimos años las empresas han tenido una mayor participación, lo que indica que las empresas están generando valor a través de estos tipos de software. En la Gráfica se observa el incremento de participación de las empresas en el último periodo.

Ilustración 24: Desarrolladores de Software libre y de fuente abierta



Fuente: Ordoñez, 2012; UNU-MERIT, 2006.

Al inicio del movimiento de Software libre, los principales usuarios que acogieron estas tecnologías fueron los estudiantes universitarios que entraron en la dinámica propositiva y proactiva que impulsó a la industria del software y la posicionó en el mercado. Con el transcurso del tiempo se consolidando un ecosistema en el que los principales agentes que participan en el proceso de desarrollo de software han sido: los autores individuales, empresas, universidades, fundaciones, grupos de desarrollo y otras organizaciones. Los cuales se describirán a continuación.

- **Autor individual**

Hay dos tipos de usuarios relacionados con el proceso de desarrollo de Software libre y de fuente abierta, que clasificaré en dos grupos. Por un lado, los usuarios consumidores que son los usuarios comunes que solo usan la tecnología, mientras que por otro, los usuarios especializados y participativos, a este tipo de usuarios, Von Hippel los clasifica como usuario innovador.

En el primer caso, el tipo de usuario consumidor se refiere a usuarios ordinarios que adquieren y usan los productos; estos han sido estudiados frecuentemente, para entender el proceso de aceptación de una innovación en el mercado, es entonces que se aplica el término de Difusión de la innovación/tecnología (Echeverría, 2013).

Rogers identifica claramente las fases de la difusión en los usuarios, desde el conocimiento de la innovación, la persuasión a la que es sometido, la decisión de probarla, la su implementación y finalmente, la confirmación de su funcionalidad; y en todo ese proceso, la aportación que hace este usuario consumidor a la innovación es informando, comentándola, elogiándola, contribuyendo a difundirla, etcétera, haciendo suyas esas propuestas innovadoras, aunque su labor se resume en hacerle propaganda a la innovación, sin darle un valor agregado a la tecnología, pero sin esta difusión la innovación no tiene impacto económico (Rogers, 2003).

En el segundo caso, el tipo de usuario innovador desarrolla o modifica productos, ya que los productos comerciales no son exactamente lo que requiere, por lo que recurren a la adaptación y creación, a este proceso se le conoce como democratización de la innovación, porque las empresas manufactureras ya no son las principales fuentes de innovación de producto (Von Hippel, 2005).

Este último tipo de usuarios usualmente comparten sus modificaciones y sus aportaciones impactan positivamente, en muchas ocasiones con mayor efectividad que las contribuciones de grandes empresas, puesto que los usuarios innovadores pueden hacer muchas modificaciones a los productos y satisfacer necesidades altamente heterogéneas de la sociedad (Von Hippel, 2005), tal como pasa con el Software libre y de fuente abierta.

La edad de los desarrolladores ha evolucionado a través del tiempo, se puede observar en la Tabla. Entre los años de 1950 y 1985, el 62% de desarrolladores tenían menos de 21 años y tan solo el 4.8% tenía más de 30 años, lo volvió al movimiento fresco y renovado. A pesar de que en los primeros periodos, el principal grupo de desarrolladores fueron menores de 21 años, que constituyeron una masa crítica para la tecnología, en los años posteriores formarían parte de los grupos de mayor edad. Sin embargo la tasa de desarrolladores en el 2002, fue de casi una cuarta parte del total, indicando que las generaciones jóvenes siguen integrándose, para seguir siendo una parte importante del SLFA.

Tabla 9: Edades de desarrolladores de SLFA

EDAD	1950-1985	1986-1990	1991-1995	2002
10-21 AÑOS	62.1%	55.0%	50.8%	24.3%
22-25 AÑOS	11.3%	24.4%	25.7%	32.4%
26-30 AÑOS	21.0%	12.2%	12.4%	27.0%
MAYORES DE 30 AÑOS	4.8%	8.4%	11.0%	16.2%

Fuente: UNU-MERIT, 2006

- **Grupos de desarrollo (Comunidades):**

Comúnmente los usuarios se integran o crean comunidades de desarrollo organizadas, las cuales pueden desarrollar consistentemente el software libre y de fuente abierta. Las comunidades pueden entenderse como agrupaciones de personas que interactúan, tienen vínculos entre ellas y además un espacio común, que en este caso es virtual.

La comunidad virtual surge cuando en la red un grupo de personas realiza discusiones públicas por un tiempo que lleva a crear redes entre las personas a través del ciberespacio. Las comunidades virtuales, también llamadas comunidades online o Usenet¹¹, han tenido un crecimiento rápido desde su surgimiento, ya que no tienen fronteras bien definidas y pueden ser dispersas geográficamente (Ninova, 2008).

La interacción entre usuarios en la red es un fenómeno común, que ha sido promovida por las nuevas herramientas de comunicación, que dan origen al concepto de software social, para referirse precisamente a las herramientas de interacción virtual. El internet permite una comunicación flexible, lo que posibilita la formación de equipos que colaboran individualmente desde distintos lugares y tiempos, y así mejora el alcance de las redes de colaboración.

La colaboración de estas comunidades debe establecer reglas de colaboración, para que puedan ser capaces de gestionar tanto la corrección de errores como dar seguimiento a las versiones. De esta manera las aportaciones individuales se pueden guiar para construir un proyecto en conjunto. Algunos ejemplos de comunidades de desarrollo son: Linux Kernel, Apache, MySQL, PHP, WordPress, GNU, X.org, Window Maker, Sugar, KDE, Gnome, Enlightenment, entre otras.

- **Empresas:**

La participación de las empresas en el desarrollo de Software libre y de Fuente abierta tiene una tendencia de crecimiento, ya que las empresas prefieren desarrollar software híbrido, entre lo privativo y libre. El software que suelen desarrollar es principalmente Software de Fuente Abierta, que cumple estas características descritas (OCDE, 2009).

Empresas actualmente consolidadas como desarrolladoras de software privativo están explorando estrategias de desarrollo del Software libre y de fuente abierta como Google, IBM,

¹¹ Es el acrónimo de Users Network, Red de usuarios en español.

Nokia, Microsoft, entre otras. Por un lado, debido a la amplia demanda de aplicaciones de software por adopción de Tecnologías de la Información y por otro lado por las tendencias tecnológicas como cómputo en la nube, el internet de las cosas, Big data y aplicaciones móviles.

- **Universidades**

Principalmente dentro de las universidades surgen muchas iniciativas relacionadas con el software libre y de fuente abierta, aunque es en menor grado que las universidades tomen una postura activa en el desarrollo de este software. Sin embargo, las universidades suelen participar en proyectos de gran relevancia como en el Proyecto Open Source Virtual Reality (OSVR), en el que participan al menos 20 universidades de EUA, Polonia, Alemania, Italia, Canadá, España, Reino Unido y Austria.

- **Fundaciones, ONGs:**

La UNESCO (United Nations Educational, Scientific and Cultural Organization) ha desarrollado software para beneficiar en áreas estratégicas tales como la manipulación y recuperación de información, minería de datos y estadísticas. La UNESCO reconoce la importancia del desarrollo del software libre y de fuente abierta, actualmente la UNESCO tiene un portal de Software Libre y de Fuente Abierta¹² (UNESCO, 2016)

El portal fue lanzado en el 2001, considerándose a la UNESCO como pionera en la promoción de este tipo de software para participar en las sociedades de la información y el conocimiento. La UNESCO reconoce (UNESCO, 2008):

- i) El software juega un papel crucial en el acceso a la información y el conocimiento;
- ii) Los diferentes modelos de software, incluyendo software privativo, de fuente abierta y el software libre, tienen muchas posibilidades para aumentar la competencia, el acceso de los usuarios, la diversidad de opciones y permitir que todos los usuarios desarrollen soluciones que mejor responden a sus necesidades;
- iii) El desarrollo y uso de las normas abiertas, compatibles, no discriminatorias para tratamiento de la información y el acceso son elementos importantes en el desarrollo eficaz de infoestructuras;

¹² Conocido por su nombre en inglés como Free and Open Source Software Portal

- iv) La proximidad de las comunidades al desarrollo de software tiene un gran potencial para contribuir a poner en práctica el concepto de sociedades del conocimiento;
- v) El software libre y de código abierto (FOSS) es modelo proporciona herramientas y procesos interesantes con la que la gente puede crear, intercambiar, compartir y explotar el software y el conocimiento eficiente y efectivamente;
- vi) El FOSS puede jugar un papel importante como un instrumento práctico para el desarrollo, lo convierten en un componente natural de los esfuerzos de desarrollo en el contexto de la Objetivos de Desarrollo del Milenio (ODM);
- vii) El apoyo constante juega un papel importante en el éxito y la sostenibilidad de soluciones FOSS;
- viii) Todas las opciones de software deben basarse en la capacidad de solución para lograr el mejor retorno de la inversión en tecnología.

En mayor o menor medida otras organizaciones como la NASA, Wikipedia, WikiLeaks, Greenpeace han contribuido a la consolidación de este sector de la industria del software.

2.5 Modelos de desarrollo de software libre y de software de fuente abierta

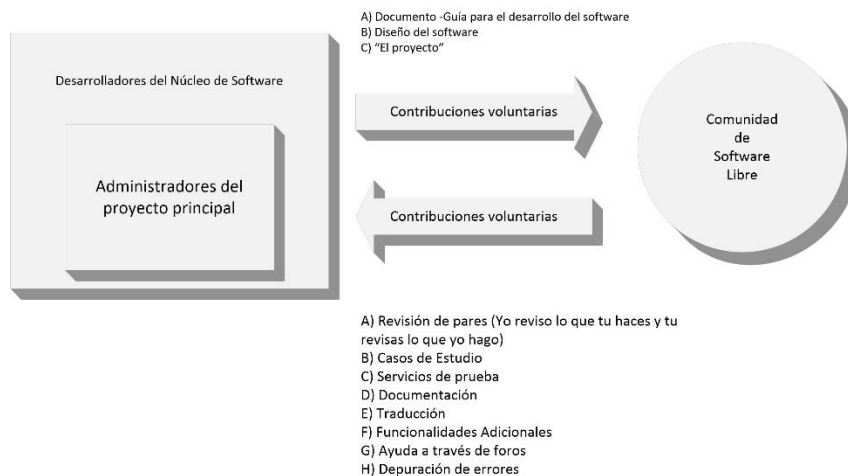
La división del proyecto en módulos en el proceso de desarrollo permite que las tareas de Software libre y de fuente abierta se puedan realizar de manera simultánea por una gran cantidad de agentes (von Hippel y von Krogh, 2003) que colaboran con su trabajo individual y que en conjunto conforman el proyecto completo.

En algunos casos, hay organizaciones o usuarios que están a cargo del proceso, sin embargo generalmente no existe una estructura jerárquica definida y se suelen emplear esquemas organizacionales horizontales y se apoyan de las comunidades software libre en el proceso. Por lo que el conocimiento y experiencia de comunidades de desarrollo se aprovecha para nuevos proyectos.

Las colaboraciones entre un proyecto de software libre y una comunidad de software libre usualmente son voluntarias, pero se tienen definidas las funciones de la cooperación. Ambas parte

trabajan por un bien común y en la mayoría de los casos no hay dinero de por medio. En este tipo de desarrollo, las comunidades son las que gestionan y dirigen el proyecto.

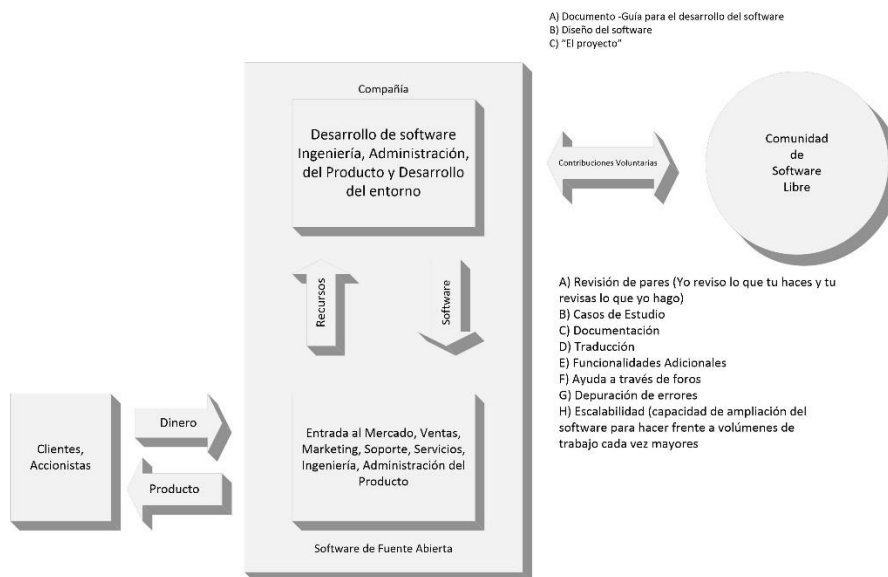
Ilustración 25: Proceso de desarrollo de software libre



Fuente: Elaboración propia basada en Ordoñez, 2009

A diferencia del software libre, los proyectos de software de fuente abierta suelen ser gestionados por empresas, con la participación voluntaria de comunidades de software libre y usuarios desarrolladores. Sin embargo este tipo de software está principalmente orientado a solucionar problemáticas del mercado buscando una retribución monetaria.

Ilustración 26: Proceso de desarrollo del software de fuente abierta



Fuente: Elaboración propia basada en Ordoñez, 2009

Las organizaciones que desarrollan Software Libre y de Fuente Abierta se benefician directa e indirectamente de estos repositorios. Los beneficios que reciben las comunidades y empresas se pueden reconocer a partir del conjunto de rutinas organizativas y procesos planteados por Zahra y George, enfocadas en el proceso de creación nuevos producto que provocan la mejora de sus capacidades organizativas y de sus capacidades tecnológicas (Flor, 2011).

CAPITULO 3: CARACTERISTICAS DEL PROCESO DE DESARROLLO DE SOFTWARE LIBRE Y DE FUENTE ABIERTA

3.1 Entradas del modelo de innovación abierta en el proceso de desarrollo de software libre y de fuente abierta

El conocimiento generado en el proceso de desarrollo de nuevos productos suele diseminarse con facilidad dentro y fuera de los límites de la empresa u organización. Si bien, el conocimiento por si mismo tiene características no rivales y no excluyentes que posibilitan que cualquier persona pueda hacer uso de éste o que pueda generar el mismo conocimiento por otros medios. La diseminación de conocimiento es un proceso natural, debido al aumento en la movilidad de trabajadores capacitados, al no tomarse en cuenta las ideas generadas en el interior de la empresa, o al incrementarse las capacidades de los proveedores (Chesbrough, 2003).

Entonces nos encontramos con dos factores importantes que intensifican la diseminación del conocimiento en el software: 1) la protección de los Derechos de Propiedad Intelectual (DPI) del software es débil; y 2) el entorno digital en el que se desenvuelve el software es altamente dinámico. Por ello, en la era digital, las empresas han tenido que evolucionar para adaptarse a las nuevas condiciones del mercado y crear nuevos modelos de negocio que les permita mantenerse vigentes, tal como iTunes, Netflix, Spotify, YouTube, entre otros.

El conocimiento tecnológico se encuentra implícito en el producto o servicio que se desarrolla, en la mayoría de los casos se requiere un proceso de decodificación, que pudiera llamarse ingeniería inversa, para descubrir el proceso inventivo y adquirir el conocimiento tecnológico implicado en la invención. En el caso del software, no solo han surgido nuevos modelos de negocio, sino también nuevos modelos de gestión de la tecnología tal como lo manifiesta el proceso de desarrollo de software libre y de fuente abierta, que promueve la diseminación del conocimiento tecnológico como una base de su paradigma de desarrollo.

De esta manera, otras empresas, organizaciones e individuos pueden aprovechar el conocimiento tecnológico generado y divulgado como código fuente de un programa, para crear nuevas soluciones. Las empresas innovadoras pueden divulgar su código fuente para crear masa crítica y posiblemente establecer un estándar en el sector, mientras que las empresas seguidoras pueden aprovechar el conocimiento tecnológico, asimilarlo, transformarlo y explotarlo, y así crear nuevo conocimiento (Kim, 2001).

Las empresas buscan mejorar sus ventajas competitivas mediante la introducción de invenciones al mercado que sean nuevas o significativamente mejoradas y que sean capaces de resolver alguna problemática de la sociedad. Durante proceso de desarrollo de nuevos productos o servicios, las empresas esperan recibir un beneficio económico, organizacional o de posicionamiento, cuando invierten en Investigación y Desarrollo (I+D), por lo que se debe tener modelos eficientes que puedan transformar los recursos de la organización en soluciones tecnológicas de alto valor para el mercado (OCDE, 2003; Feldman, 2004).

En el caso del software propietario, la I+D ocurre al interior de la organización con contribuciones externas limitadas, por el contrario, el software libre y de fuente abierta aprovecha las contribuciones externas a su I+D interno, incluso empresas u organizaciones trabajan en conjunto, aunque no se puedan apropiar del conocimiento generado, lo que reduce costos (West & Gallagher, 2006).

Ilustración 27: Flujos de conocimiento en tres modelos de I+D de software



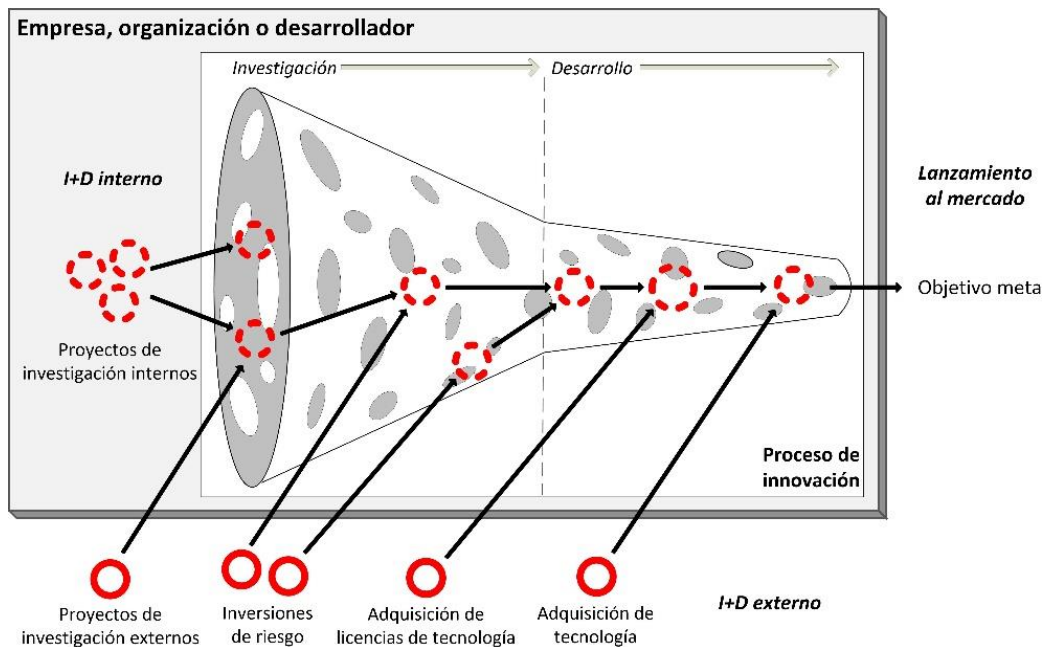
Fuente: Elaboración propia tomado de West y Gallagher, 2006.

La contribución y gestión del conocimiento en la Investigación y Desarrollo puede ser una forma de clasificar la innovación abierta del software, ya que el software privativo se puede representar por el modelo de innovación propietaria o innovación cerrada, mientras que el software libre y de fuente abierta puede representarse por esquemas como I+D compartida y *Spinout*, que permiten por un lado trabajar en conjunto con otros agentes, como también beneficiar a la comunidad con los conocimientos generados.

En el proceso de desarrollo de software libre y de fuente abierta, se deben considerar los factores externos involucrados para evaluar la importancia de cada uno de ellos. La empresa, organización o desarrollador individual inicia el desarrollo de un nuevo proyecto a partir de una base tecnológica propia, su I+D interno, sin embargo durante el proceso de desarrollo se nutre principalmente de factores externos.

Por ello, se identifican y agrupan las principales entradas del modelo de innovación abierta del proceso de desarrollo de software libre y de fuente abierta, tal como: repositorios que contienen el código fuente de la solución de muchos proyectos o financiamiento colectivo, mediante la aportación de capital de numerosas personas que comparten el riesgo.

Ilustración 28: Entradas en el proceso de desarrollo del software



Fuente: Elaboración propia

Los elementos de I+D externo son entradas al modelo de innovación abierta que complementan la base tecnológica interna durante el proceso de desarrollo, ya sea en una etapa inicial formando una base de tecnología externa o en una etapa intermedia a avanzada reforzando el proyecto y mejorando sus características técnicas. El software libre y de fuente abierta se nutre del trabajo previo de otras personas por medio de repositorios, blogs, tutoriales, foros, entre otros, y además recibe entradas financieras que sostienen el proyecto, siendo el financiamiento colectivo, el principal modelo de financiamiento en este tipo de software.

- **Base de tecnología externa.**

En el proceso de desarrollo de software libre y de fuente abierta, otros proyectos previos pueden ser la base tecnológica externa de los nuevos proyectos, por lo que las bases del proyecto están respaldadas por uno o varios proyectos funcionales que resuelven alguna problemática. Por lo que surgen repositorios de código fuente de software que contienen proyectos que pueden ser adaptables, reutilizables, actualizables, escalables y autoexplicables (Dávila, Nuñez, Sandia, & Torrén, 2006). Algunos repositorios de artículos científicos de acceso abierto promuevan la divulgación libre del conocimiento, sin embargo la divulgación del conocimiento de los repositorios de software es más efectiva, ya que el conocimiento codificado tiene aplicación inmediata y se puede conocer de manera abierta la forma en la que se construyó el proyecto a través de su código fuente.

Los repositorios de software agrupan los códigos fuente de proyectos desarrollados para cumplir funciones específicas y su disponibilidad permite, por un lado, usarlos y aprender de ellos, y por otro, modificarlos y adaptarlos a nuevas condiciones para resolver otros problemas aprovechando el trabajo previo de otras personas. Algunos repositorios conocidos como <http://sourceforge.net> y <http://directory.fsf.org/> son catálogos de código fuente de proyectos de software, y algunos otros repositorios como GitHub, además de ser catálogos de códigos fuente, permite el control de versiones de los proyectos. Los usuarios disponen de una base tecnológica de códigos fuente de proyectos de software implementados y probados, que podrían utilizar, adaptar o mejorar, es decir adecuarse a las necesidades particulares del problema, dependiendo de la libertades expresadas por las licencias que utilicen los programas del repositorio.

Los repositorios de software libre y de fuente abierta permiten concentrar y organizar el conocimiento tecnológico previo de muchas personas, por lo que algunos gobiernos han implementado este esquema para aprovechar el trabajo de proyectos anteriores y de esta manera reducir los tiempos y costos de los nuevos proyectos, como es el caso del Gobierno de Canarias y la Universidad de la Laguna de España, la Junta de Andalucía de España, el Ministerio del Poder Popular para la Educación Universitaria, Ciencia y Tecnología de Venezuela, el Gobierno de Chile y la Subsecretaría de Tecnologías de Gestión Jefatura de Gabinete de Ministros de Argentina.

Mientras que en México, no han sido suficientes los esfuerzos para implementar repositorios públicos de software libre y de fuente abierta, ya que los esfuerzos recientes solo han

sido por establecer un Repositorio Nacional de Acceso Abierto a Recursos de Información Científica, Tecnológica y de Innovación, de Calidad e Interés Social y Cultural (DOF, 2014). Los únicos esfuerzos relacionados directamente son los de homologar estructura y contenidos de los sitios web del Gobierno Federal que permitió emplear software libre con nuevos proveedores de servicios a costos muy reducidos (Saldaña, 2016).

Tabla 10: Repositorios de software públicos

Nombre	Origen	Descripción
OpenPYME	La Agencia Canaria de Investigación, Innovación y Sociedad de la Información del Gobierno de Canarias y la Universidad de la Laguna	Es un catálogo de Software Libre donde se recopilan, de forma categorizada, productos sólidos y fiables que pueden incorporarse en cualquier ámbito productivo de una empresa, mejorando así su gestión y competitividad gracias a la inclusión de herramientas TIC.
Repositorio de software	Junta de Andalucía	Es un repositorio que incluye software desarrollado a medida que hace uso tanto de librerías y plataformas libres, como comerciales. La junta de Andalucía no proporcionará soporte técnico de instalación o configuración, pero se proporcionará el nombre de la empresa desarrolladora que podría ofrecer el servicio de soporte.
Repositorio Nacional de Aplicaciones	Ministerio del Poder Popular para la Educación Universitaria, Ciencia y Tecnología de Venezuela	El Repositorio Nacional de Aplicaciones (RNA), busca ser un espacio colaborativo de referencia, donde se encuentran y promueven aplicaciones, herramientas y proyectos en TI, que son desarrollados en Software Libre bajo estándares abiertos, de utilidad e interés para la Administración Pública y las comunidades organizadas.
Repositorio Software público	Gobierno de Chile	Es una iniciativa de la Unidad de Modernización del Estado y Gobierno Digital de Segpres, para fomentar el desarrollo de aplicaciones reutilizables entre instituciones públicas, constituyendo un espacio de intercambio y colaboración.
Repositorio de Software libre argentino	Subsecretaría de Tecnologías de Gestión Jefatura de Gabinete de Ministros. Argentina	Se promoverá el uso del Repositorio de Software Público Argentino dentro de las dependencias del Estado Nacional y se brindará asistencia técnica a los organismos nacionales, provinciales y municipales que así lo requieran. Se creará la plataforma online para compartir aplicación y proyectos. Se avanzará en aplicaciones para la gestión del conocimiento por comunidades técnicas de Software Público. Se emprenderán acciones en pos de normar sobre la Licencia de Software Público.

Fuente: Elaboración propia basada en: Oficina de Software Libre de la Universidad de la Laguna-España, 2016; Junta de Andalucía- España, 2016; Ministerio del Poder Popular para la Educación Universitaria, Ciencia y Tecnología- Venezuela, 2016; Unidad de Modernización del Estado y Gobierno Digital- Chile, 2016; la Alianza para el gobierno abierto- Argentina, 2016.

- **Adquisición de Tecnología: Compra de APIs**

En el proceso de desarrollo de software libre y de fuente abierta, los proyectos se nutren de conocimiento tecnológico como las APIs que son Interfaces de programación de aplicación o paquetes tecnológicos que cumplen una función particular, están constituidos por un conjunto de subrutinas y métodos que ofrecen una biblioteca para que esta pueda ser integrada por otro programa. De esta forma los desarrolladores se benefician de las funciones programadas de por otros desarrolladores, sin tener que programar toda la aplicación desde cero y aprovechar un bloque de programa que cumple una función, aunque las APIs son módulos estandarizados y tiene el inconveniente de perderse cierta flexibilidad en el proyecto.

Otra fuente de tecnología externa son las librerías o bibliotecas que son un conjunto de herramientas de software que cumplen funcionalidades específicas, las cuales se emplean continuamente en determinados programas. De esta forma se simplifica el proceso de desarrollo de software, por la reducción de tiempo y esfuerzo para escribir y depurar código. Finalmente, durante el proceso de desarrollo, los desarrolladores cuentan con conocimiento técnico específico divulgado mediante cursos, tutoriales, blogs y foros, siendo los foros el espacio o plataformas de discusión donde la comunidad de programadores construyen nuevo conocimiento, tal como StackOverflow.

- **Financiamiento**

Por otro lado, además de las entradas de conocimiento tecnológico, se requiere también financiamiento que sostenga e impulse a los proyectos de software. El financiamiento en el software varía mucho y depende de cada caso, algunos proyectos emplea solo desarrolladores voluntarios y los pocos costos relacionados están cubierto por donativos, mientras que otros proyectos son financiados en su totalidad

En el caso de proyectos de software libre y de fuente abierta, el financiamiento externo puede considerarse como patrocinio, sin embargo este apoyo no siempre es desinteresado, por lo que se deben explotar las formas de financiar proyectos de software. Una de estas formas de financiamiento, es mediante un gobierno o institución pública que tienen objetivos particulares como científico, social o promoción de estándares.

Otras formas de financiamiento retribuyen beneficios relacionados con los productos del programa, ya sea en forma de libros, hardware, discos con programas para que los inversionistas

perciban algún tipo de beneficio. Sin embargo, uno de los principales medios de financiamiento del software es a través del financiamiento colectivo, conocido comúnmente como crowdfunding, que mediante plataformas web promociona proyectos que solicitan fondos ya sea de forma de donación, inversión o préstamo. La característica principal de este modelo es el masivo número de inversores que comparten el riesgo. Algunas plataformas de Crowdfunding son:

Tabla 11: Plataformas de financiamiento colectivo.

#	SITIO	VOLUMEN (DÓLARES)	RANKING ALEXA EU	TARIFA	OBSERVACIONES
1	go fund me	\$470 M	282	5%	Más de \$2 mil millones recaudados para los proyectos personales. Se aplican tarifas de procesamiento es de 2.9% + \$0.30
2	Kick starter	\$444 M	199	5%	Proyectos personales no son permitidos, solo creativos. Se aplican tarifas de procesamiento de entre el 3 al 5%.
3	Indiegogo	-	642	5%	Tarifa de procesamiento de 3% y una tarifa de \$25 por banca internacional
4	teespring	-	1,263	10%+	Sitio de crowdfunding para playeras. Las tarifas varían de acuerdo a las camisetas seleccionadas para la venta.
5	patreon	-	893	5%	Solo proyectos creativos. La tarifa de procesamiento es de 4% adicional.
6	YouCaring.com	-	2,932	5%	Se sugiere una tarifa de 5% a los donadores. La tarifa de procesamiento es de 2.9% + \$0.30.
7	Crowdrise	-	4,004	5%	Las cuentas gratuitas tienen una tarifa de 5%, las de cuenta de pago un 3%. La tarifa de procesamiento es de 2.9% + \$0.30.
8	DonorsChoose.org	-	5,639	15%	La tarifa es opcional de 15% para apoyar a DonorsChoose.org. La donación es 100% deducible de impuestos.
9	KIVA	-	6,008	15%	La tarifa es de 15% y sugiere hacer una campaña a micro-prestamistas. Tarifa de procesamiento de 2.9% + \$0.30
10	Giveforward	-	18,041	5%	Tarifa de 5% se cobra a los creadores de la campaña. Tarifa de procesamiento es de 2.9% + \$0.50

Fuente: elaboración propia basado en www.crowdfunding.com, 2014.

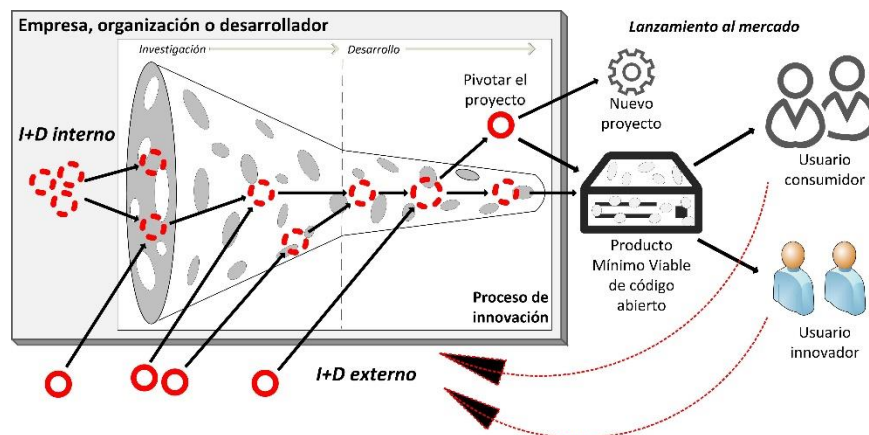
3.2 Salidas del modelo de innovación abierta del proceso de desarrollo de software libre y de fuente abierta

El desarrollo del software es incremental, ya que cada sección o bloque del programa debe probarse individualmente y en conjunto. Las pruebas se desarrollan junto a los requerimientos, es decir los planes de prueba parten desde el diseño del sistema. En este proceso se contempla en algunas ocasiones la prueba de aceptación, cuando se desarrolla el software para un cliente en específico, llamada también prueba Alfa.

Por otro lado, cuando un software se va a comercializa a gran escala, se realiza una prueba llamada prueba Beta, que permite a un número determinado de clientes puedan probar una versión preliminar del software, en condiciones normales de uso real y retroalimentar a los desarrolladores sobre los errores que no habían sido identificados. En algunos casos, se realizan versiones Beta adicionales (Sommerville, 2005).

En este caso, los usuarios consumidores toman un papel importante, que en este caso un grupo de ellos contribuye al desarrollo del proyecto con notificación de errores y preferencias de uso, que permite analizar las cualidades técnicas y dirección del proyecto. El lanzamiento de la versión Beta debe realizarse cuando el software cumpla las condiciones básicas sin importar que no esté completamente acabado. Reid Hoffman, fundador de LinkedIn, afirmaba que “si, cuando lanzas el producto, no te avergüenzas de este, es que lo has lanzado demasiado tarde” (Ismail, Malone, & Van Geest, 2016).

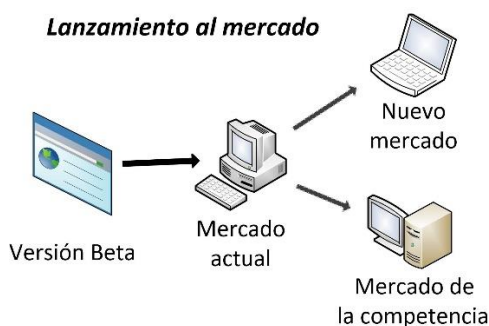
Ilustración 29: Lanzamiento de la versión Beta



Fuente: Elaboración propia, basada en Chesbrough, 2003

El mismo procedimiento puede llevarse a cabo por un grupo de usuarios con conocimientos técnicos, usuarios innovadores, que pueden proporcionar retroalimentación más precisa sobre el software. Sin embargo, sus contribuciones son de un nivel técnico más profundo, ya que tiene la capacidad de reconocer y corregir por sí mismo los errores o propone mejoras en las funcionalidades del software, depende de la comunidad a cargo del proyecto el modelo de gestionar las contribuciones externas.

Ilustración 30: Pivoteo de solución de software

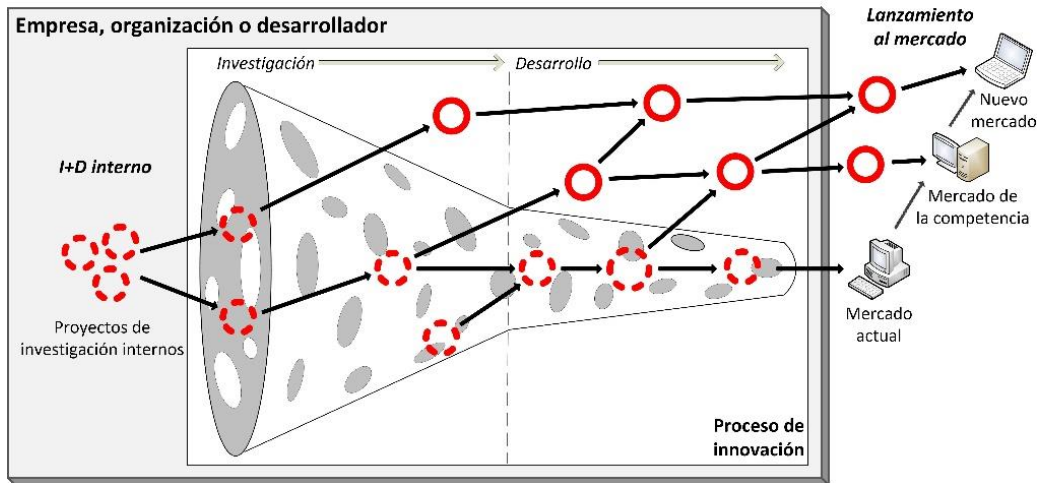


Fuente: Elaboración propia.

En este punto, requiriendo el menor esfuerzo posible se tiene una versión de software que cumple con características funcionales mínimas para ser mostrado y utilizado como un producto final, aunque aún puede ser modificado o mejorado. En otras palabras, la versión Beta también podría identificarse con el término de Producto Mínimo Viable (PMV), introducido por Eric Ries en su metodología de Lean Startup, en 2011 (Osma Aponte, 2015).

A partir del análisis de la retroalimentación de la versión Beta, los encargados del desarrollo del proyecto de Software Libre o de Fuente Abierta tienen la decisión de continuar con el mismo objetivo del programa o incluso, pivotar, redirigir sus esfuerzos hacia una nueva dirección, ya sea en el mismo mercado, mercado de la competencia o incluso crear un mercado nuevo, como lo han hecho empresas como: Youtube, sitio de vídeos, comenzó siendo un sitio de citas online; Instagram, plataforma de fotografías, nació como una plataforma para calificar y recomendar lugares geolocalizados; o incluso se pivota los modelos de negocios como lo hizo Nokia, que originalmente era una fábrica de papel, posteriormente de goma, cable, hasta que entraron a la industria electrónica.

Ilustración 31: Salidas del modelo de innovación abierta



Fuente: Elaboración propia basada en Chesbrough, 2003.

Las estrategias que una organización utiliza le permiten impactar y tomar una cuota del mercado, pero los modelos de negocio respaldan la forma de subsistencia, por lo que las organizaciones deben definir adecuadamente el modelo de Negocios que empleará. El modelo de negocios es el medio para que la empresa crea, entregue o capture valor, por ello las versiones Beta sirven para reconocer si la tecnología que se está desarrollando tendrá éxito comercial en el futuro.

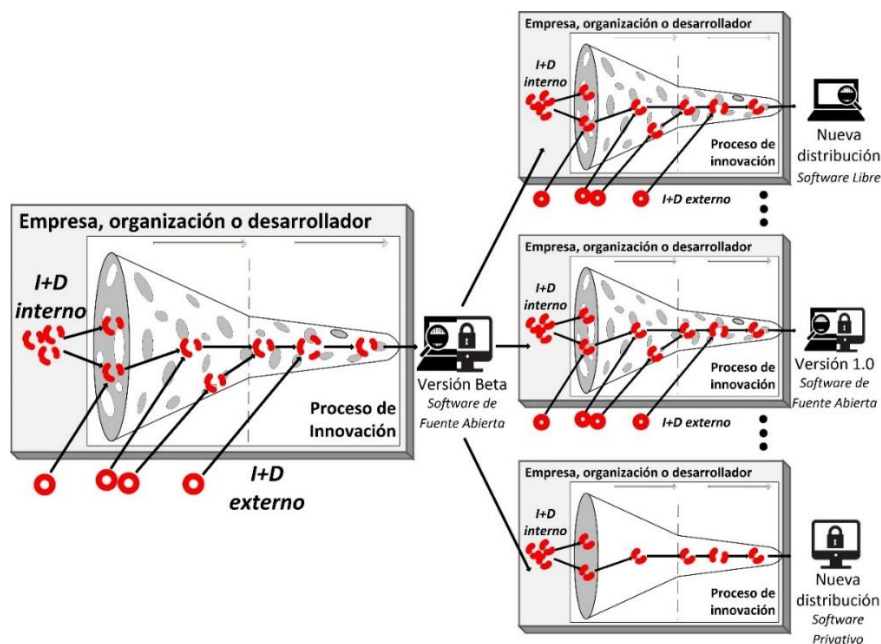
En el caso del Software Libre y de Fuente Abierta, los modelos de negocio pueden diferir si se trata de uno u otro de los tipos de software, por ejemplo el modelo de negocio más simple, se basa en donaciones, como lo hace Wikipedia y algunas otras comunidades, pero Cygnus Solutions fue de las primeras empresas en demostrar que se podía hacer negocio del software libre, ofertando el servicio de soporte técnico a otras organizaciones.

Sun Microsystems, con Java, ha hecho negocio de forma diferente, convirtió su software en un formato libre, compatible con múltiples plataformas, lo que le permitió extender su tecnología globalmente, aunque la empresa no pierde el control de la dirección de desarrollo, teniendo una ventaja para ofrecer más y mejores complementos con costo para su software estrella.

En el caso de MySQL y Sleepycat, presenta una licencia dual, y los clientes pueden usar su software libremente, incluso aplicar un esquema en el que cooperan y compiten con el proveedor (Kim, 2001) manteniendo la fusión de códigos con Open Source, pero si quieren integrarlo a un software privativo deberán comprar una licencia específica. De esta forma se beneficia la empresa

que está a cargo del software con licencia dual, ya que a cambio de su producto tiene una versión mejora de su producto o sino ingresos por la compra de una licencia.

Ilustración 32: Versiones posteriores del Software de Fuente Abierta



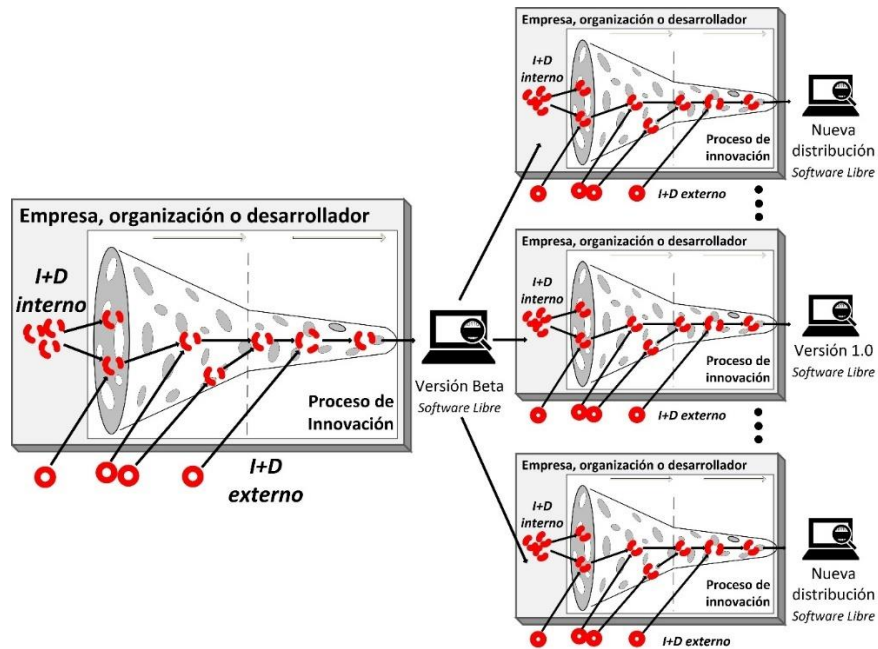
Fuente: Elaboración propia.

Sin embargo, lo más común es encontrar organizaciones que ofrecen gratuitamente un producto básico y restringido, mientras que el producto mejorado tiene un costo; o simplemente desde el inicio se vende el código fuente. Lo importante de este proceso es la transferencia rápida y cabal de conocimiento, con remuneración para el creador. En el desarrollo de los productos y servicios hay una tendencia para que sean altamente personalizables. Esta diversidad de productos expande el mercado y más desarrolladores se integran al mercado para crear accesorios de tecnología modular.

La tendencia de las empresas desarrolladoras de software es emplear esquemas híbridos que incluyan tanto a software propietario, como a software libre en sus productos (OCDE, 2009), lo que abre una oportunidad de crecimiento de software de fuente abierta y la posibilidad de generar valor y ganancias con este software. Ya que el software de fuente abierta, el proyecto tiene características tanto privativas como libres, que le permiten gestionarse de la misma manera en las versiones posteriores, aportar conocimiento técnico a proyectos de software libre o incluso en

algunos modelos de negocio, permitir que una nueva versión del proyecto se transforme a una versión de software propietario (licencia dual).

Ilustración 33: Versiones posteriores del Software libre



Fuente: Elaboración propia.

Por el contrario, debido a que un software libre se rige bajo las 4 libertades, todas sus derivaciones seguirán siendo libres. Por ello, el software libre tiene limitaciones para hacer negocios, pero representa un esquema ideal para afrontar problemáticas comunes, a través de la participación voluntaria de numerosas personas.

Aunque el software de fuente abierta representa un esquema que combina modelos de desarrollo libres y privados, que le permiten adaptarse a las condiciones del mercado y principalmente al modelo de negocios de las empresas. El Software de fuente abierta es un esquema de desarrollo que se puede promover fácilmente a través de la iniciativa privada.

3.3 Proceso de aprendizaje. Imitación- innovación

El Internet junto con otras tecnologías de la información y comunicación han creado un nuevo ecosistema económico en la red, que se caracteriza por ser rápido, global, digital y democrático, que plantea nuevas reglas de negocios. Estas nuevas condiciones posibilitaron que pequeñas empresas pudieran crecer y consolidarse como Google o Amazon, mientras que otras empresas que no se adaptaron a los cambios terminaron por desaparecer; Schumpeter se refería a este proceso como “transmutación de valores” (Echeverría, 2013). Particularmente en el software, la transmutación de valores es constante por el surgimiento de innovaciones disruptivas que tienen impacto mundial como Uber o Airbnb.

Los proyectos de software son una fuente constante de conocimiento técnico, por ello las empresas, organizaciones o desarrolladores individuales pueden aprovechar y apropiarse del conocimiento tecnológico de otras personas. Es decir, el conocimiento explícito plasmado en el código fuente de proyectos previos se combina con el conocimiento tácito de programadores para construir un nuevo proyecto con mejores características técnicas, que también se comparte para que pueda ser utilizado por otros. El código fuente divulgado es asimilado mediante un proceso de interiorización más breve, ya que el conocimiento codificado tiene algunos comentarios aclaratorios, una estructura definida y es autoexplicativo para las personas que tienen cierto dominio de programación.

El aprovechamiento del conocimiento externo depende de la capacidad de absorción de conocimiento de los miembros participantes del proyecto, para algunos autores como: Kim (1997) para crear nuevo conocimiento se requiere de la capacidad para aprender y solucionar problemas; mientras que para Lane, Koka y Pathatk (2006) para explotar conocimiento, se requiere de un proceso de aprendizaje que tenga la habilidad de identificar y asimilar conocimiento del entorno; sin embargo Cohen y Levinthal (1990) identifican la importancia de reconocer el valor de la información nueva, para asimilarla y posteriormente aplicarla.

Por ello, es indispensable que las empresas, organizaciones o desarrolladores individuales fortalezcan sus capacidades de absorción de conocimiento, a partir de sus procesos y rutinas organizativas, las cuales pueden agruparse en dos bloques: I) La capacidad de absorción potencial que se encarga en adquirir e identificar conocimiento externo, para ampliar su base de

conocimientos y poderse adaptar al dinamismo del entorno; y II) La capacidad de absorción realizada que requiere las capacidades de transformación explotación y pueden influir mediante una innovación, garantiza la explotación del conocimiento, a diferencia de la capacidad de absorción potencial (Zahra, George, 2002).

En primer lugar, las organizaciones requieren de la habilidad para reconocer el valor de información externa nueva, para asimilarla y aplicarla con fines comerciales, a partir del conocimiento previo de la organización y de los individuos que la conforman (Cohen- Levinthal, 1990). En el caso del software libre y de fuente abierta que transfiere código fuente, no se requiere codificación, ya que el código fuente por sí mismo es conocimiento codificado.

En el proceso de creación de nuevos productos, las organizaciones que generan software libre y de fuente abierta emplean sus rutinas y procesos propios para mejorar la eficiencia de sus resultados. Por lo que se requiere que las empresas tengan una avanzada capacidad de absorción del conocimiento, que es descrita como los procesos de aprendizaje fundamentales para identificar, asimilar y explotar el conocimiento del entorno (Lane, Koka y Pathatk, 2006). Si bien hay varias definiciones para describir este proceso de absorción del conocimiento, se empleará la de Zahra y George (2002) que la definen como un conjunto de rutinas organizativas y procesos para que las empresas adquieran, asimilen, transformen y exploten el conocimiento.

Las rutinas organizativas y procesos constan de cuatro etapas: en la etapa de adquisición, las organizaciones deben identificar y adquirir nuevo conocimiento externo con intensidad, velocidad y dirección; en la etapa de asimilación, se debe analizar, procesar, interpretar y comprender la información obtenida; en la tercera etapa, transformación, las empresas mejoran o desarrollan nuevas rutinas a partir de una combinación con el conocimiento previo; finalmente la última etapa, permite que las organizaciones perfeccionen, extiendan y aprovechen sus competencias y desarrollen nuevas para explotar el conocimiento generado.

En el proceso de desarrollo de software libre y de fuente abierta, la etapa de adquisición de conocimiento, donde las organizaciones tienen a su disposición repositorios o catálogos de códigos fuente de muchos proyectos, lo que les permite filtrar, dirigir la búsqueda de acuerdo a los objetivos particulares de los proyectos y adquirir conocimiento útil con intensidad y velocidad de acuerdo a los requerimientos del software.

Durante todo el proceso de desarrollo de software se realizan búsquedas para adquirir nuevo conocimiento externo, pero cada conjunto de conocimiento adquirido sigue una siguiente etapa. En la etapa de asimilación de conocimiento, las organizaciones cuentan a su disposición cursos, tutoriales y blogs que muestran algunos ejemplos que ayudan a comprender el conocimiento tecnológico adquirido. Aunque las plataformas de discusión se convierten en plataformas de intercambio de conocimiento que permite construir una comunidad a través de estos espacios digitales.

A partir de la recolección y comprensión de una base de conocimiento, se continúa con la siguiente etapa: la transformación. En esta etapa, las organizaciones tienen posibilidad de algún proyecto de fuente abierta, para modificarlo y adaptarlo a sus necesidades, en esta etapa, otras organizaciones o empresas ofrecen sus servicios técnicos para apoyar en el proceso de transformación de conocimiento.

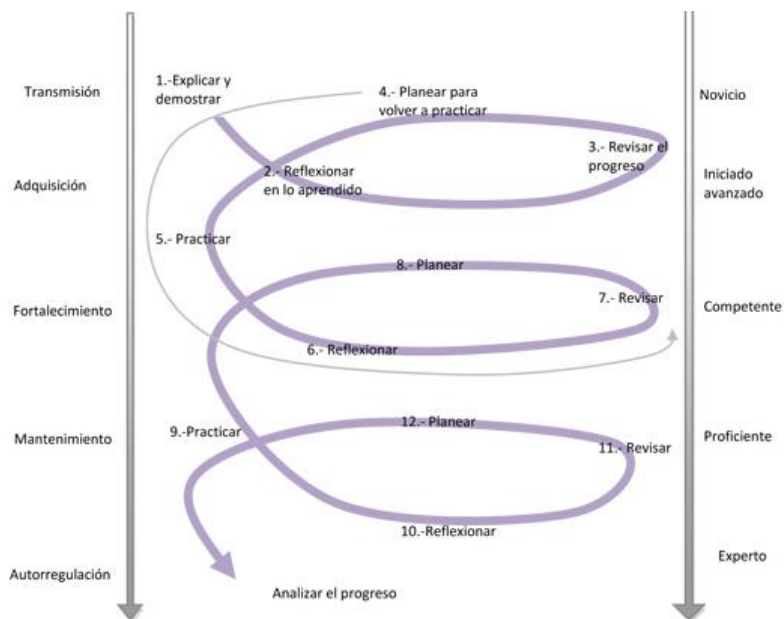
Finalmente en la cuarta etapa, de explotación, las organizaciones integran las nuevas competencias y son capaces de extender su campo de acción, su propuesta de valor es mayor, sus productos se perfeccionan y pueden recibir retroalimentación del mercado cuando son usados (Flor, 2011). En este ciclo de aprendizaje, las organizaciones que desarrollan Software Libre y de Fuente Abierta mejoran sus capacidades tecnológicas y de gestión tecnológica.

A partir del código fuente de proyectos previos, las empresas y organizaciones que desarrollan software libre y de fuente abierta pueden replicar la tecnología inmediatamente, es decir imitar la tecnología para posteriormente lograr una adaptación o mejorar el proyecto en poco tiempo. El proceso de absorción de conocimiento para este tipo de software es más efectivo, ya que en el caso del software privativo, las empresas u organizaciones tienen que decodificar la tecnología externa para interiorizarla, aunque en este proceso solo se logra interpretar la tecnología externa en nivel general.

De esta manera, la capacidad de absorción de conocimiento de las empresas y organizaciones que desarrollan software libre y de fuente abierta es más efectiva, ya que es posible interpretar a fondo la tecnología previa y comparar funciones de diferentes proyectos, lo que permite obtener mejores opciones técnicas para construir una solución más eficiente, que mejora las capacidades tecnológicas de los agentes involucrados en el proceso de desarrollo de software.

El proceso de desarrollo de software implica una evolución individual del conocimiento, que puede explicarse a partir de la espiral de conocimiento propuesta, desde un nivel de conocimientos nulos hasta un nivel de conocimiento experto. El aprendizaje es un proceso de etapas cíclicas que permiten aumentar las capacidades tecnológicas y de innovación de los desarrolladores. Estas etapas de aprendizaje son: 1) Explicar y demostrar, 2) Reflexionar en lo aprendido, 3) Revisar el progreso, 4) Planear para volver a practicar, 5) Practicar, 6) Reflexionar, 7) Revisar, 8) Planear, 9) Practicar, 10) Reflexionar, 11) Revisar, 12) Planear, 13) Analizar el progreso.

Ilustración 34: Modelo de espiral práctica



Fuente: Parsloe y Wray, 2002; Tejada 2007.

Las etapas de la espiral del conocimiento se repiten, pero el nivel de complejidad y dominio del conocimiento aumenta. El proceso de desarrollo de software, los agentes involucrados corresponden principalmente a un grupo de conocimientos técnicos más avanzados que reconocen sus símbolos a partir de un conocimiento base, por lo que pueden interpretar conocimiento más complejo. Lo que significa que la transferencia de tecnología, el conocimiento no se transfiere solo en manuales y guías, sino en niveles más avanzados donde se crea nuevo conocimiento tecnológico.

Si bien, el principal medio de transferencia de conocimiento es a través del código fuente, otro medio fundamental para transferir el conocimiento del software es a través de la discusión y el análisis colectivo, empleando plataformas como blogs y foros. Estas plataformas se convierten en

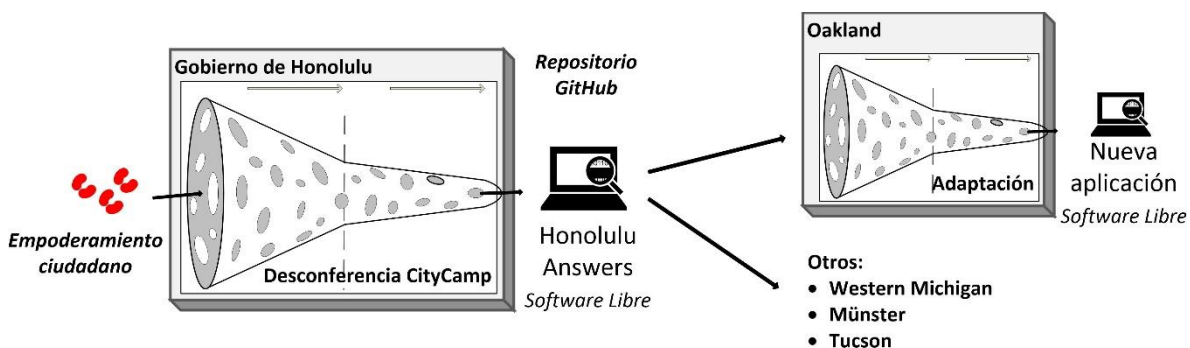
un medio para crear comunidades digitales que no tienen fronteras y que pueden construir conocimiento social utilizando el internet como una zona de desarrollo próximo.

Por otro lado, los agentes novatos tienen a su alrededor elementos que les permiten desarrollar capacidades tecnológicas desde cero, mediante algunos recursos que apoyen a elevar su nivel de comprensión y así puedan colaborar en un nivel más complejo. Es decir, en este tipo de software el conocimiento no se baja al nivel de información y datos, sino que son los actores que aumentan su nivel de comprensión, como un pensamiento complejo (Morin, 1990).

Algunos ejemplos de este proceso de imitación-innovación, inician a partir de una solución funcional que se adapta fácilmente para replicarse en otros entornos tal como la plataforma de Honolulu Answers que permite a los residentes de la ciudad conocer sobre los servicios gubernamentales mediante un buscador amigable que responde específicamente a las preguntas que los usuarios escriben en un recuadro.

En 2011, el Hawaii buscaba que Honolulu fuera un gobierno más abierto mediante una desconferencia¹³ enfocada en la innovación y colaboración para gobiernos municipales, llamada CityCamp. La desconferencia no está programada para una audiencia pasiva, el contenido es creado y organizado por los participantes y coordinado por facilitadores, que proporciona un excelente formato para la creatividad y busca crear comunidades locales que desarrollen aplicaciones que apoyen a hacer ciudades más abiertas y amigables (Hibbets, 2013).

Ilustración 35: Proceso de imitación-innovación



Fuente: Elaboración propia.

¹³ Una desconferencia, también llamado No-Congreso o conferencia de espacio abierto, es una conferencia en la que los propios participantes y asistentes toman un papel más participativo y activo.

En la desconferencia CityCamp de Honolulu, se empoderó a los ciudadanos para construir un proyecto en Code for America sobre una plataforma interactiva de preguntas y respuestas que sería llamada Honolulu Answers, que sería divulgada en GitHub. El éxito de esta aplicación y la posibilidad de acceder al código fuente permitió replicar y adaptar la solución a otras ciudades como Oakland, Western Michigan, Münster, Tucson entre otras (Code for America, 2016).

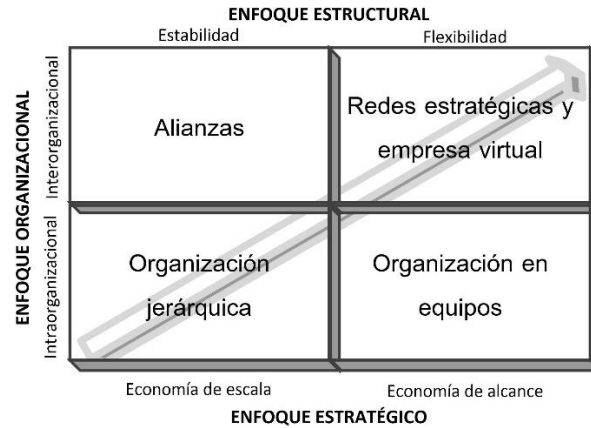
3.4 Gestión de la colaboración en el proceso de desarrollo de software libre y de fuente abierta

La cobertura del internet se ampliará en los próximos años, aumentando la masa crítica de este medio y permitiendo las personas accedan a la información y el conocimiento de todo el planeta. Se espera que el tráfico del Internet también aumente, el pronóstico para el periodo de 2014 a 2019 es que casi se cuadruplica la cobertura de internet y además que el tráfico por persona crezca de 6 Gigabytes a 18 Gigabytes. Es decir que habrá mayor acceso a internet y las personas harán un mayor uso de éste (CISCO, 2015).

Un crecimiento en el número de personas que usa internet significa en un primer momento un aumento en la demanda de soluciones de software, aunque también representa la oportunidad de aprovechar las contribuciones de una cantidad mayor de personas. La estructura de colaboración del software libre y de fuente abierta es horizontal y participan muchos actores que conforman redes estratégicas. Esta red de agentes tiene un enfoque estructural más flexible y un enfoque organizacional con mayores vínculos al exterior.

Las condiciones ideales para una buena gestión de conocimiento es mediante una cultura organizacional colaborativa (Rodríguez, 2006) misma que es impulsada por el Software libre y de fuente abierta. La sociedad se beneficia de tecnología y conocimiento especializados, a un bajo costo. Las organizaciones y usuarios desarrolladores, optimizan su gestión organizacional y la gestión del conocimiento, lo que tiende a desarrollar redes estratégicas entre los agentes involucrados.

Ilustración 36: Evolución de las modelos organizacionales



Fuente: Elaboración propia tomada de Montoya Restrepo & Montoya Restrepo, 2012

El trabajo colaborativo de los desarrolladores de software libre y de fuente abierta puede ser analizado acuerdo al modelo de conversión y creación de conocimiento, propuesto por Nonaka y Takeuchi (1995). Como se había analizado, el primer proceso de transferencia de conocimiento entre organizaciones o individuos, parte de una externalización, es decir, el primer agente plasma su conocimiento tácito en código fuente o diagramas, mientras que el segundo agente que lo absorbe, interioriza el conocimiento externo, lo recibe, lo comprende y junto con su conocimiento previo incrementan su conocimiento tácito (Kim, 2001).

Una de las ventajas de esta transferencia de conocimiento, en el software libre y de fuente abierta, es que no siempre es necesario decodificar el conocimiento implícito, ya que muchos de los productos resultantes de esta colaboración siguen siendo de código abierto. En el proceso de desarrollo de software libre y de fuente abierta colaboran en una gran red de agentes, por ello le dan gran importancia a la creación de redes, así como del conocimiento. De esta forma, los agentes involucrados en el proceso de desarrollo conforman redes de conocimiento (Robert, Yoguel & Erbes, 2007).

La colaboración entre agentes debe gestionarse con la finalidad de que las aportaciones puedan ser consideradas en las nuevas versiones del software, por lo que se requiere de aplicaciones o plataformas que puedan gestionar las versiones del software, incluso tener la posibilidad de regresar a versiones anteriores. Estos sistemas de control de versiones permiten incluso a los desarrolladores individuales mantener el control durante el proceso de desarrollo.

El proyecto se gestiona y se mejora a partir de las aportaciones, si la nueva versión del proyecto implica cambios significativos, ya sean radicales o semiradicales, cambia una unidad en la versión, es decir de la Versión 1-0 a la 2.0 y cuando se tratan de cambios incrementales, la nueva versión simplemente cambia una décima, es decir de 1.0 a 1.1. Es decir, un cambio del primer número implica grandes cambios o mejoras, mientras que un cambio en números subsecuentes implica pequeños cambios, correcciones o una fase de pruebas.

Por otro lado, los sistemas de control de versiones proporcionan un sistema de gestión del conocimiento eficiente, ya permite respaldar continuamente los avances en determinados momentos para evitar pérdidas y además tener la posibilidad de regresar a alguna de las versiones anteriores. La evolución del proyecto puede observarse a través de una lista de cambios históricos de la versión original del proyecto. Además algunos sistemas de control de versiones permiten compartir los códigos, por lo que otros usuarios pueden replicar inmediatamente su solución, adaptarla y mejorarla.

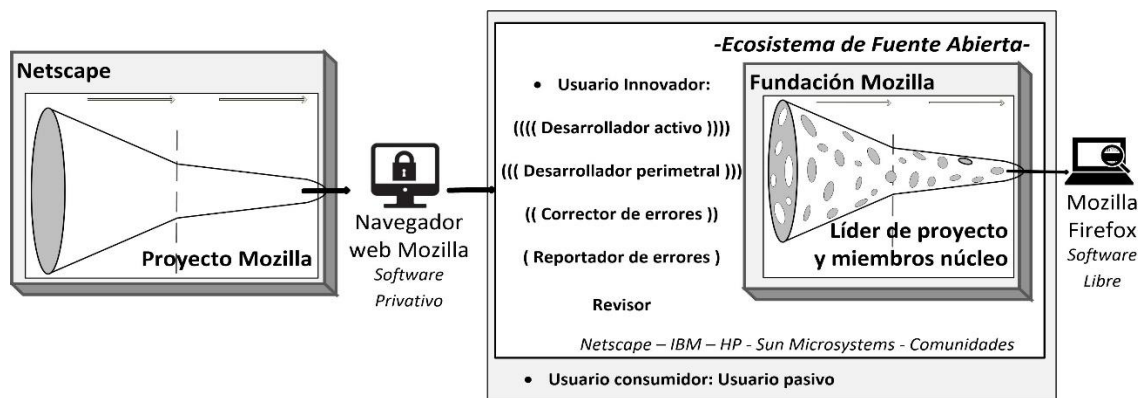
Sin embargo, el trabajo colaborativo del software libre y de fuente abierta requiere la creación de nuevos modelos de gestión de la colaboración, que a diferencia de los modelos de gestión del conocimiento, este resuelve directamente una problemática mediante la participación colectiva. El proyecto completo consta de la colaboración de numerosas personas que aportaron su trabajo, aunque para gestión dicho trabajo se requiere un sistema de gestión de la colaboración.

Un ejemplo de gestión de la colaboración es el proyecto del navegador web Mozilla creado como respuesta al navegador Internet Explorer de Microsoft en 1998 por Netscape. Cinco años más tarde, Netscape tomó el rol de patrocinador y defensor de la comunidad de software libre y de fuente abierta, que consiguió un proyecto del software propietario migrará a un proyecto de software libre (West & Gallagher, 2006).

El proyecto Mozilla requirió seguimiento por lo que se creó una comunidad abierta, en el 2003, la Fundación Mozilla, para gestionar las aportaciones que mejoraban las características o agregaban nuevas funcionalidades. En el 2013, la comunidad era más grande que cualquier empresa y se demostró que empresas pueden obtener beneficios de la colaboración en proyectos de código abierto.

En este proyecto del navegador web de Mozilla, contribuyeron miembros que no son participantes directos del proyecto, desde patrocinadores hasta aportaciones técnicas de organizaciones, empresas, académicos, aficionados individuales y otros grupos interesados. IBM, HP y Sun Microsystems destinaron ingenieros de software para que trabajaran en este proyecto, ya que estaban interesados un navegador que fuera compatible con sus respectivos sistemas y aunque las empresas no pudieron proteger el conocimiento generados, se beneficiaron de los productos del trabajo colaborativo. Este modelo de colaboración se basa una comunidad a cargo del proyecto dentro del ecosistema social de comunidades y desarrolladores de software libre y de fuente abierta, es un modelo de colaboración común, que incluso se emplea en otros entornos, ya que el modelo de colaboración de Wikipedia está basado en el Software libre.

Ilustración 37: Proceso de desarrollo del Proyecto Mozilla



Fuente: Elaboración propia basada en Kilamo, 2012.

Este modelo describe a la comunidad como una cebolla, es decir: en el corazón del modelo se encuentra el líder del proyecto que suele ser el fundador o el miembro más activo; posteriormente un grupo núcleo de miembros, donde inician comúnmente 7 y terminan creciendo a 10 a 15 miembros; en el siguiente nivel se encuentra un grupo de desarrolladores activos, descritos anteriormente como usuario innovador, con distintos roles como desarrollador activo, desarrollador periférico, desarrollador que corrige bugs, usuario que reporta bugs, lector; y finalmente el usuario pasivo, descrito anteriormente como usuario consumidor.

Otro modelo de colaboración es a través de retos cívicos, en donde alguna comunidad o institución abre una convocatoria para que desarrolladores individuales o agrupados propongan soluciones. De esta manera la colaboración se gestiona como una lluvia de idea, recibiendo todo tipo de proyectos que serán sometidos a concurso para elegir a las mejores. Este proceso fue llevado

a cabo por la comunidad de Codeando México, que realizó una convocatoria para encontrar aplicaciones de software libre o de fuente abierta que pudieran cumplir las mismas funciones que la aplicación de un contrato de 115 millones de pesos.

En el 2013, la Cámara de diputados de México adquirió una aplicación que apoyaría a los diputados, que tenía un costo que superaba los 115 millones de pesos, con un acuerdo multianual que finalizaba en agosto de 2015. Las principales funciones de la aplicación eran monitorear y dar seguimiento a la información generada en el congreso, que son consideradas funcionalidades sencillas para un desarrollador común, por lo que la comunidad de Codeando México lanzó un reto para desarrollar alternativas que incluso se mejoraran las funcionalidades de la aplicación contratada.

Codeando México convocó mediante el hashtag #app115 o la etiqueta “Derrocando a la Mexican Tech Mafia” y logró la participación de 172 personas, de las cuales se eligieron a cinco aplicaciones finalistas, que fueron presentadas en un evento llamado “Codeando México: App de Código libre. Hacia un Congreso Digital”. Finalmente el ganador del reto de Codeando México se le otorgo un medio de 11 mil pesos y un iPad mini, que es solo significativo a comparación de costo del contrato de 115 millones de pesos. Es decir, el premio de este reto fue menor al 0.01% del costo del contrato de la aplicación comprada.

Este es un ejemplo claro de las ventajas de construir soluciones en una red de colaboración, ya que surgieron muchas propuestas de mejor calidad y en un periodo menor a dos semanas, lo que ocasionó la cancelación del contrato de la aplicación de los 115 millones. Las capacidades de las comunidades y de los desarrolladores individuales permitieron construir soluciones altamente heterogéneas en poco tiempo, lo que plantea la posibilidad de resolver algunas problemáticas sociales a partir del empoderamiento y capacidades tecnológicas y de innovación de los programadores participativos o usuarios innovadores.

3.5 Aprendizajes y expectativas del proceso de desarrollo del software libre y de fuentes abierta

A lo largo de esta investigación se ha mostrado algunas de las características del proceso de desarrollo de software libre y de fuente abierta, explicado bajo los términos de la innovación abierta,

sin embargo no se puede generalizar las características identificadas, aunque se puede aprender e identificar las buenas prácticas de los modelos y estrategias empleadas, para replicarlas en otros entornos ajenos al software.

De manera de conclusiones previas, la divulgación del código fuente junto con los archivos que posibilitan el funcionamiento de las aplicaciones, permite en un primer momento aprender, replicar y construir proyectos más eficaces. Es decir, un componente importante en el proceso de desarrollo es el aprovechamiento del conocimiento previo, divulgado abiertamente en repositorios de código abierto.

Los repositorios de software son una base de conocimiento que puede contener: *aplicaciones*, que son proyectos comúnmente probados; *librerías*, que son un conjunto de herramientas estandarizadas empleadas continuamente, que ahorra tiempo y trabajo en el desarrollo; y *APIs* que son herramientas de uso específico desarrolladas por otros y que pueden integrarse a los proyectos.

Por otro lado, los tutoriales y foros están en proceso de evolucionar a una etapa de mayor integración debido a las nuevas herramientas. Por ejemplo, los foros de discusión como *stackoverflow*, en donde cualquier usuario puede preguntar o responder alguna pregunta, se construye nuevo conocimiento, sin embargo se repiten comúnmente las mismas preguntas, por lo que herramientas como Watson de IBM podría comprender y agrupar las preguntas para que antes de publicar una pregunta, el sistema pueda mostrar preguntas relacionadas que pudieran responder la duda del usuario.

La gestión de la colaboración de los usuarios innovadores y de los usuarios consumidores permite aprovechar el trabajo y talento de muchas personas para construir un proyecto en conjunto. El software libre se gestiona comúnmente a partir de comunidades o algunas empresas que guían y seleccionan las aportaciones de los desarrolladores o a través de una convocatoria abierta para seleccionar las mejores aplicaciones que solucionen de mejor manera la problemática. Mientras que el software de fuente abierta permite esquemas disruptivos que han sido empleados por varios años como innovación pública o innovación abierta privada. Recientemente, en otros escenarios se han empleado estas estrategias como la innovación abierta privada que promovido la empresa Tesla al divulgar y permitir el uso de parte de su tecnología patentada.

Finalmente, uno de los aprendizajes más importantes del proceso de desarrollo del software libre y de fuente abierta, radica en la recomendación de probar en el mercado un prototipo funcional de la tecnología para aprender de las condiciones reales y determinar si es necesario abandonar y replantear el proyecto. En este proceso, el usuario innovador aporta sus conocimientos técnicos para mejorar la solución, pero el usuario consumidor también tiene un papel importante en la retroalimentación de los proyectos.

Esta investigación no es completamente concluyente, es simplemente un bosquejo del proceso de desarrollo de software libre y de fuente abierta, ya que pocos de los fenómenos disruptivos de los últimos años han sido documentados. Por lo que, se pueden desprender nuevas líneas de investigación que podrían detonar en nuevos modelos de gestión del conocimiento, tecnología, innovación o colaboración.

CONCLUSIONES

Las computadoras han evolucionado para ser más potentes y accesibles, que permitió el esparcimiento de sistemas mínimos de computadora que ahora abundan a nuestro alrededor. Sin embargo, debido a la extensión de aplicaciones y usos de los sistemas mínimos de computadora, el concepto de computadoras se ha ido difuminando por lo que es necesario redefinir al conjunto de tecnologías que se encuentran en nuestro entorno.

La evolución del cómputo ha sido exponencial, desde los orígenes de la computadora electrónica que requería más de 167 metros cuadrados hasta computadoras integradas en la ropa, electrodomésticos o dispositivos móviles, aunque la evolución más significativa dentro del cómputo es sobre la capacidad de duplicar la potencia de computadoras cada 18 meses, descrita en la ley de Moore, pero apenas estamos iniciando lo que será la siguiente revolución del cómputo, a través de la computación cuántica.

La industria del software tiene menos de 50 años, el internet, 25 años, pero han sido protagonistas del desarrollo económico actual, que la caracterizan por ser una industria altamente dinámica, por lo que ha sido la base de esquemas económicos disruptivos como el software libre y el software de fuente abierta, ya que comparten el código fuente de sus programas. Estos tipos software no tienen un gran impacto en las masas, pero dominan entornos más especializados que requieren mayor seguridad y robustez.

Hay una amplia gama de aplicaciones de software que se clasifican de acuerdo a sus características y usos, como software de aplicación, de sistema o de programación. El software es un producto por sí mismo, se puede vender o comprar, ya sea software empaquetado o hecho a la medida, excepto el software embebido que no se puede vender independientemente, sino como parte de un producto.

El software libre se originó bajo un concepto idealista de compartir libremente el conocimiento de la humanidad, por lo que su código fuente siempre será libre, en cambio el software de fuente abierta que es un modelo híbrido del software libre y del software privativo. El software libre no es gratuito, pero su venta directa dificulta el cumplimiento de sus libertades, por lo que términos prácticos termina siendo gratuito, en cambio el software de fuente abierta posibilita la creación de otros modelos de negocios a partir del este software.

La protección de la propiedad intelectual del software es muy débil en la mayoría de los países, por lo que surgen las licencias de software basadas en derechos de autor como alternativa, que son contratos del desarrollador o distribuidor con el usuario. Aunque muchas instituciones públicas o privadas han propuesto licencias propias, las licencias de Creative Commons han destacado.

El proceso de desarrollo del software libre y de fuente abierta se basa en la construcción de proyectos basados en otros trabajos previos y en la participación de agentes externos, además de la posibilidad de derivar el producto principal en otras soluciones distintas, por lo que resulta natural explicar este proceso bajo el modelo de innovación abierta, incluso una versión con mayor interdependencia que podría describirse bajo el concepto de innovación abierta 2.0.

En el proceso de desarrollo de software libre y de fuente abierta participan dos tipos de usuarios: el usuario consumidor es quien usa el producto resultante, pero también puede contribuir probando versiones Beta de programas y retroalimentando a los desarrolladores, mientras que el usuario innovador es quien tiene conocimientos técnicos y colabora para construir el software. El principal agente que participa en el desarrollo colaborativo son programadores individuales, es decir, los usuarios innovadores.

El conocimiento tecnológico del software se divulga intensamente debido a la débil protección de la propiedad intelectual y a las características propias del entorno digital, por lo que compartir el código fuente es una estrategia que se adapta a las condiciones del mercado y plantea nuevos modelos de negocio. El software libre y de fuente abierta permite diversas estrategias de innovación como la innovación pública, la innovación abierta privada y la innovación de fuente abierta.

Por otro lado, el código fuente es equivalente a las recetas, diagramas y otros archivos que ayudan a entender, replicar y modificar la tecnología plasmada, por lo que se pueden satisfacer con rapidez las necesidades altamente heterogéneas de la sociedad, mediante adaptaciones de otros proyectos de software libre y de fuente abierta.

El código fuente del software es conocimiento codificado, de un nivel superior de los datos y la información, por lo que de primer momento es también superior a guías o manuales. Por lo que

la divulgación del conocimiento codificado, se lleva a cabo en una red de personas especializadas, mediante símbolos y códigos por lo que la asimilación del conocimiento es casi instantánea.

La construcción colaborativa del software libre y de fuente abierta permite construir más soluciones, con mejores características con un menor costo y tiempo, que puede desarrollarse a partir de una comunidad o empresa. Los repositorios permiten agrupar conocimiento tecnológico de muchos proyectos, conocer la evolución en su desarrollo y de sus proyectos derivados.

Deben lanzarse versiones básicas y funcionales de los programas lo más pronto posible para tener una retroalimentación en condiciones reales, con la finalidad de detectar errores y analizar si conviene continuar desarrollando el proyecto o cambiar su objetivo, es decir pivotar.

El software libre y de fuente abierta permite desarrollar capacidades tecnológicas en personas y organizaciones, además de la posibilidad de construir proyectos más ambiciosos y con mayor rapidez, mediante la pequeña colaboración de muchos usuarios. Por ello es recomendable impulsar ecosistemas basados en este software, ya que permite una estrategia aprovecha conocimiento previo para afrontar mayores retos y generar soluciones de software constantemente, por lo que su ciclo de vida es corto.

Las características benéficas y dinámicas del software, posicionan a la industria del software como una industria innovadora, particularmente el sector de la industria dedicada al desarrollo de software libre y de fuente abierta, ya que poseen cualidades óptimas para desarrollar capacidades tecnológicas y de innovación. La gestión de la innovación en el software libre y de fuente abierta es más flexible y rápido, ya que se basa en estructuras organizacionales horizontales y en la participación de múltiples actores. Lo que implica que este modelo de desarrollo es un ejemplo para otras industrias.

El software libre y de fuente abierta permite la comprensión, manipulación y adaptación del código fuente, que a su vez fomentan las capacidades de desarrollo de software. La importancia de desarrollar este tipo de capacidades se basa en los beneficios implicados como empleos mejor remunerados, la posibilidad de solucionar problemas específicos de la sociedad (mediante el software), impulsar empresas en una industria de alto valor agregado. Por su lado, las organizaciones se benefician con la alta capacidad de innovación, mejora en sus capacidades organizativas y tecnológicas, además de una eficiente gestión del conocimiento.

El software libre y el software privativo son incompatibles, pero el software de fuente abierta surge como alternativa, ya que es compatible con ambos tipos de software, permitiendo esquemas de negocios híbridos para las organizaciones, que van desde donaciones, servicio de soporte técnico, venta de complementos, venta directa de código fuente, versión Premium con costo o por licencia dual (los clientes deben pagar si requieren que sea privativa la fusión su software con el de fuente abierta).

Los principales desarrolladores de SLFA son jóvenes, sin embargo es indispensable promover de manera sistemática el fomento de capacidades de innovación y desarrollo de software en la población en general, desde edades tempranas.

El factor clave en el proceso de desarrollo de software libre y de fuente abierta es la óptima gestión del conocimiento, que permite aprovechar conocimiento y participación externa. Además los participantes del proceso de desarrollo tienen un nivel de comprensión de un mayor nivel, por lo que colaboran en un nivel más complejo con conocimiento codificado de aplicación inmediata.

En el software libre y de fuente abierta, se tiene la ventaja que usuarios innovadores, empresas u organizaciones externas pueden contribuir a mejorar la solución de software. Por lo que tienen buenos sistemas de gestión del conocimiento, llamadas sistemas de control de versiones, que permiten visualizar y controlar las modificaciones que se van realizando a lo largo del desarrollo del proyecto.

Finalmente, impulsar el software libre y de fuente abierta puede ser un potente motor para toda la economía y podría lograr una economía competitiva, por lo que debe tomarse en cuenta en los planes de desarrollo de empresas, organizaciones, regiones y países.

Referencias

- Fan, J., Stuart, G., & Yu, X. (2013). Innovation or imitation? The role of intellectual property rights protections. *Journal of Multinational Financial Management*, 23, 208-234.
- 20 Minutos. (6 de Junio de 2015). *20 minutos*. Recuperado el 6 de Junio de 2015, de La NASA cambia Windows por Linux en la estación Espacial Internacional (ISS): <http://www.20minutos.es/noticia/1809627/0/nasa/sistema-operativo/windows-por-linux/>
- built with. (1 de octubre de 2016). *trends.builtwith.com*. Obtenido de <https://trends.builtwith.com/web%20server#>
- Cámara de Diputados del H. Congreso de la Unión. (2016). *Ley de la Propiedad Industrial*. Ciudad de México: Diario Oficial de la Federación.
- CENATIC. (2009). Centro Nacional de Referencia de Aplicación de las TIC basadas en fuentes abiertas. *Software de fuentes abiertas: Licencias*.
- CISCO. (1 de agosto de 2015). *CISCO*. Recuperado el 1 de agosto de 2015, de [www.cisco.com](http://www.cisco.com/web/solutions/sp/vni/vni_forecast_highlights/index.html): http://www.cisco.com/web/solutions/sp/vni/vni_forecast_highlights/index.html
- Coar, K. (4 de junio de 2015). *The Open Source Definition*. Obtenido de www.opensource.org
- Cobo, J. C. (2009). El concepto de tecnologías de la información. Benchmarking sobre las definiciones de las TIC en la sociedad del conocimiento. *Zer - Revista de Estudios de Comunicación*, 295-318.
- Code for America. (3 de noviembre de 2016). *Apps y APIs*. Obtenido de Honolulu Answer: <http://archive.codeforamerica.org/products/honolulu-answers/>
- Cohen, W. M., & Levinthal, D. A. (1990). Absorptive Capacity: A New Perspective on Learning and Innovation. *Administrative Science Quarterly*, 128-152.
- Creative Commons. (s.f.). *History*. Recuperado el 26 de julio de 2016, de [CC: https://creativecommons.org/about/history/](https://creativecommons.org/about/history/)
- Curley, M. (2016). Twelve principles for open innovation 2.0. *Nature*, 314-316.
- Daniel Caraballo, M. M. (s.f.). *Estudio del Open/Free (GNU/Linux) como plataforma de servicios de red en entornos empresariales*. Obtenido de http://www.fing.edu.uy/~asabigue/prgrado/2004eofgl/contenido/anexo2/anexo_ii_22.html
- Dávila, J. A., Nuñez, L. A., Sandía, B., & Torréns, R. (2006). Los repositorios institucionales y la presevación del patrimonio intelectual académico. *Interciencia*, Vol 31.
- Dávila, T., Epstein, M. J., & Shelton, R. (2006). *Making Innovation Work. How to manage it, measure it, and profit from it*. New Jersey: Pearson Education, Inc.
- DOF. (20 de 05 de 2014). *Diario Oficial de la Federación*. Obtenido de http://www.dof.gob.mx/nota_detalle.php?codigo=5345503&fecha=20/05/2014

- FORBES. (2015). *The World's Biggest Public Companies*. Recuperado el 6 de Junio de 2015, de <http://www.forbes.com/global2000/list/#industry:Software%20%26%20Programming>
- Foro Consultivo Científico y Tecnológico, AC. (2015). *Catálogo de programas para el fomento a la innovación y la vinculación en las empresas 2015*. Distrito Federal: FCCyT.
- Free Software Foundation. (10 de 6 de 2015). *GNU Operating System*. Obtenido de ¿Qué es el software libre?: <http://www.gnu.org/philosophy/free-sw.html>
- González, M. (14 de octubre de 2015). *centros.edu.xunta.es*. Obtenido de MAGASI: http://centros.edu.xunta.es/iesxulianmagarinos/webantiga/webs_deptos/Informatica/proxectos/magasicd/en/home_es.html
- Grove, M. (2008). *Open Innovation 2.0*. Obtenido de Technology Innovation Management Review: <http://timreview.ca/article/216>
- Hibbets, J. (2013). *The foundation for an open source city*. Raleigh, Carolina del Norte.
- IBM. (4 de mayo de 2016). *IBM pone disponible el cómputo cuántico desde la nube para acelerar la innovación*. Obtenido de Comunicados de prensa IBM: <http://www-03.ibm.com/press/mx/es/pressrelease/49671.wss>
- IBM. (5 de 09 de 2016). *IBM Quantum Computing*. Obtenido de Research IBM: <http://www.research.ibm.com/quantum/>
- Ikusimakusi. (26 de diciembre de 2012). *El semáforo de Creative Commons*. Obtenido de <http://ikusimakusi.eus/2007-2015/es/2012/el-semaforo-de-creative-commons/>
- Instituto Mexicano para la Competitividad A.C. (2014). *Los emprendedores de TIC en México: Recomendaciones de política pública para su nacimiento, crecimiento y consolidación*. Microsoft.
- Ismail, S., Malone, M. S., & Van Geest, Y. (2016). *Organizaciones exponenciales*. BUBOK PUBLISHING.
- Joskowicz, J. (2016). *//eva.fing.edu.uy*. Recuperado el 26 de julio de 2016, de Facultad de Ingeniería de la Universidad de la República - Uruguay: [https://eva.fing.edu.uy/pluginfile.php/67082/mod_resource/content/3/Historia%20Telecomunicaciones%20\(presentacion\).pdf](https://eva.fing.edu.uy/pluginfile.php/67082/mod_resource/content/3/Historia%20Telecomunicaciones%20(presentacion).pdf)
- Junta de Andalucía. (23 de noviembre de 2016). *Repositorio de Software*. Obtenido de http://www.juntadeandalucia.es/repositorio/index.jsf;jsessionId=A773C37FC466CB82EA4A56CACDF29C00?linkDummyForm:_idcl=_id113&
- Kim, L. (1997). *From Imitation to Innovation: The Dynamics of Korea's Technological Learnig*. Harvard Business School Press.
- Kim, L. (2001). La dinámica del aprendizaje tecnológico en la industrialización. *Revista internacional de ciencias sociales*, 168.
- La alianza para el gobierno abierto. (13 de noviembre de 2016). *Software público argentino*. Obtenido de <http://www.opengovpartnership.org/es/country/argentina/commitment/software-p%C3%BAblico-argentino>

- Lane, P. J., Koka, B., & Pathak, S. (2006). The reification of Absorptive Capacity: A Critical Review and Rejuvenation of the Construct. *Academy Management Review*, 833-863.
- Levy, H. P. (20 de octubre de 2015). *Smarter with Gartner*. (Gartner, Editor) Recuperado el 5 de julio de 2016, de What's New in Gartner's Hype Cycle for Emerging Technologies, 2015: <http://www.gartner.com/smarterwithgartner/whats-new-in-gartners-hype-cycle-for-emerging-technologies-2015/>
- Lima Ramos, A. (2015). Software de fuentes abiertas: el paradigma de desarrollo del futuro tecnológico. En X. Martínez Ruiz, *Infoesfera* (págs. 154-171). Ciudad de México: Colección Paideia XXI. Coordinación Editorial de la Secretaria Académica del Instituto Politecnico Nacional.
- María, F., Oltra, M., & García, C. (2011). La relación entre la capacidad de. *Revista Europea de Dirección y Economía de la Empresa*, 69-88.
- Ministerio del Poder Popular para la Educación Universitaria, Ciencia y Tecnología. Gobierno Bolivariano de Venezuela. (20 de noviembre de 2016). *Repositorio Nacional de Aplicaciones*. Obtenido de <http://repositorio.softwarelibre.gob.ve/>
- Montoya Restrepo, L., & Montoya Restrepo, I. (2012). APLICACIÓN DE LA METÁFORA BIOLÓGICA PARA EL DESARROLLO DE FORMAS ORGANIZATIVAS EN LA INTEGRACIÓN EMPRESARIAL. *Revista Facultad de Ciencias Económicas: Investigación y Reflexión*.
- NETCRAFT. (6 de Junio de 2015). *Web server developers: Market share of all sites*. Recuperado el 6 de Junio de 2015, de Netcraft: <http://news.netcraft.com/archives/category/web-server-survey/>
- NETMARKETSHARE. (2015). *netmarketshare*. Recuperado el 10 de Junio de 2015, de netmarketshare.com
- Ninova, M. G. (2008). Comunidades, software social e individualismo conectado. *Athenea Digital: revista de pensamiento e investigación social*, 299-305.
- OCDE. (2005). *Manual de Oslo. Guía para la recogida e interpretación de datos sobre innovación*. Oslo: Grupo Tragsa.
- OCDE. (2009). *Innovation in the Software Sector*. OCDE Publishing.
- Olivo, C. (1999). Breve historia de la computadora. *Revista de la universidad de Mendoza*(17).
- Open Source Initiative. (6 de Junio de 2015). *Open Source*. Recuperado el 6 de Junio de 2015, de <http://opensource.org/osd>
- Oppenheimer, A. (2014). *¡Crear o morir!: La esperanza de Latinoamérica y las cinco claves de la innovación*. Nueva York: Debate.
- Ordóñez, S. (En prensa). *El sector electrónico-informático y de las telecomunicaciones y el desarrollo en México*. México, Distrito Federal: IIEc-UNAM.
- Ordoñez, S., & Ortega, R. (2009). El capitalismo del conocimiento y el software libre y de fuente abierta: historicidad y la nueva alternativa de desarrollo para el siglo XXI. *Economía UNAM*, 113-136.

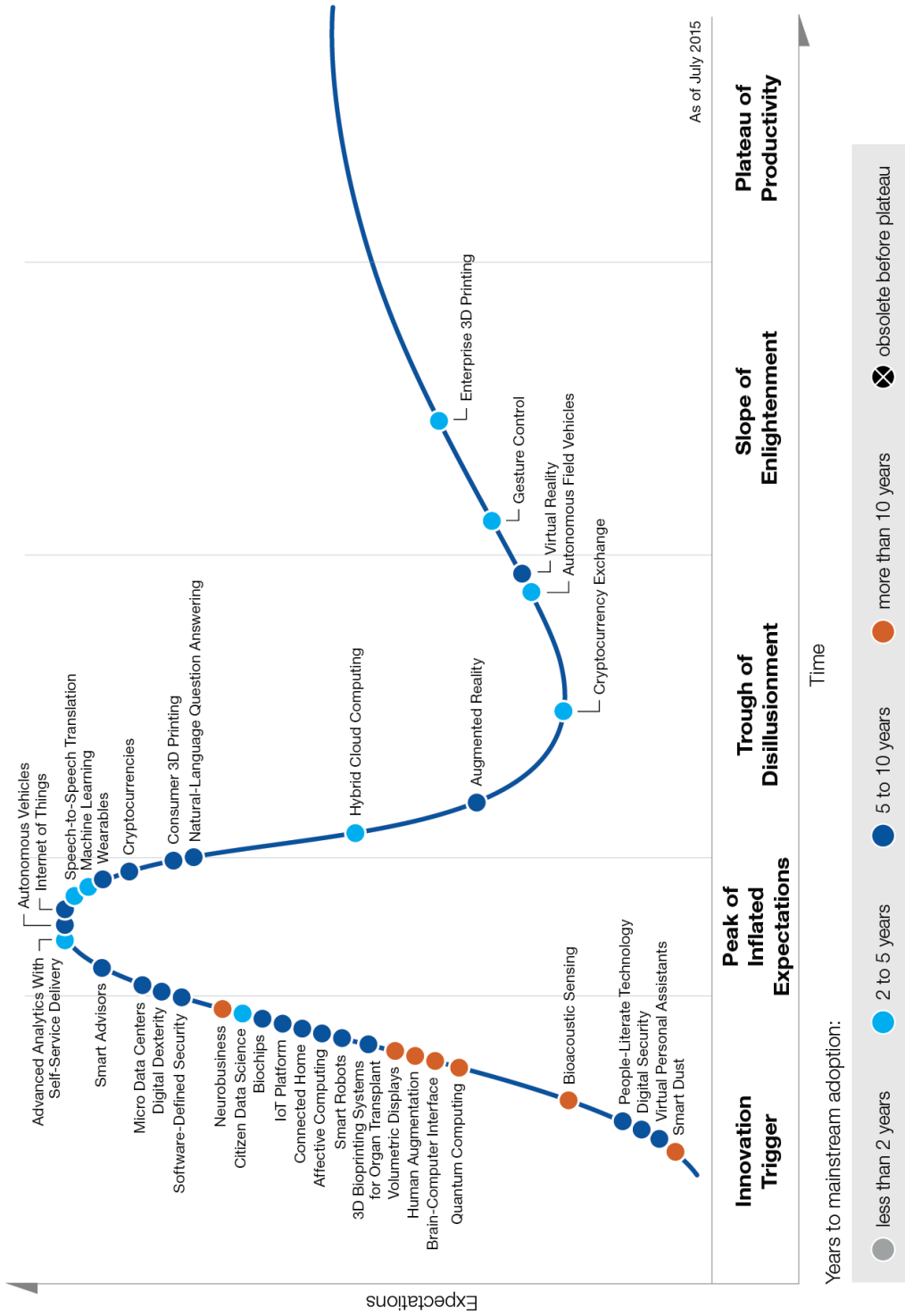
- Osma Aponte, S. A. (2015). *Análisis, diseño y desarrollo del producto mínimo viable para el videojuego "Magic Cave"*. Bogotá.
- Past, Present and Future. (2013). *International Journal of Engineering Business Management*.
- Quijano, J. (13 de noviembre de 2014). *GENBETA:dev*. Obtenido de .NET se pasa a Open Source: <https://www.genbetadev.com/actualidad/net-se-pasa-a-open-source-novedades-importantes-en-connect>
- R, R. (1992). Successful industrial innovation: critical factors for the 1990s. *R&D Management*, 221-240.
- R, R. (1994). Towards the fifth-generation innovation process. *International Marketing Review*, 7-31.
- RICYT. (2006). *Manual de Lisboa: pautas para la interpretación de los datos estadísticos disponibles y la construcción de indicadores referidos a la transición de Iberoamérica hasta la sociedad de la información*.
- Rodríguez Gómez, D. (2006). Modelos para la creación y gestión del conocimiento : una aproximación teórica. *Educar*(Núm. 37), Páginas: 25 - 39. Obtenido de <http://www.raco.cat/index.php/Educar/article/view/58019/68087>
- Rogers, E. M. (2003). *Diffusion of Innovations*. New York: Free Press.
- Romaní, J. C. (2009). El concepto de tecnologías de la información. Benchmarking sobre las definiciones de las TIC en la sociedad del conocimiento. *Zer*, 295-318.
- Salazar, M. (6 de diciembre de 2016). #App115. (A. Lima Ramos, Entrevistador)
- Saldaña, E. (28 de enero de 2016). Software libre en México. (A. Lima Ramos, Entrevistador)
- Sampedro, J. L. (2011). *Conocimiento y empresa: industria del software en México*. Ciudad de México: Editorial Plaza y Valdés S.A. de C.V.
- Sommerville, I. (2005). *Ingeniería del software*. Madrid: Pearson Education.
- Stallman, R. (2002). *Free Software, Free Society*. GNU Press.
- Thierry Bücheler, J. H. (2011). Understanding Science 2.0: Crowdsourcing and Open Innovation in the Scientific Method. *Procedia Computer Science*, 327-329.
- Übergizmo. (5 de mayo de 2016). *ubergizmo.com*. Obtenido de <http://es.ubergizmo.com/2016/05/05/ibm-ofrece-por-primera-vez-computacion-cuantica-en-la-nube.html>
- UNESCO. (2005). *Oficina de Información Pública*. Recuperado el 15 de 06 de 2015, de Las tecnologías de la información : http://www.unesco.org/bpi/pdf/memobpi15_informationtechno_es.pdf
- UNESCO. (2008). *Proprietary and Free and Open Source Software*. Paris: UNESCO House.
- UNESCO. (2016). *Communication and Information*. Obtenido de UNESCO's Free and Open Source Software Portal: <http://www.unesco.org/new/en/communication-and-information/access->

to-knowledge/free-and-open-source-software-foss/unescos-free-and-open-source-software-portal/

- UNU-MERIT. (2006). *Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies*. Netherlands.
- Velez, M., & Sicard, A. (2000). Computación cuántica: una perspectiva desde lo continuo. *Revista Universidad EAFIT*, 41-46.
- Vercelli, A. (2004). *La conquista silenciosa del Ciberespacio. Creative Commons y el diseño de entornos digitales como nuevo arte regulativo en internet*. Buenos Aires, Argentina.
- Webbink, M. (2005). Classification of (software) licenses in context of copyright according to Mark Webbink. Recuperado el 26 de julio de 2015, de https://www.redhat.com/f/summitfiles/presentation/May31/Open%20Source%20Dynamics/Troan_OpenSourceProprietyPersp.pdf
- West, J., & Gallagher, S. (2006). Challenges of open innovation: the paradox of firm investment in open-source software. *R&D Management*, 319-331.
- Žižlavsky, O. (2013). Past, Present and Future of the Innovation Process. *International Journal of Engineering Business Management*.

ANEXO 1: Ciclo de sobre expectativa de tecnologías emergentes

Emerging Technology Hype Cycle



ANEXO 2: Licencias de software libre y de fuente abierta

1. Academic Free License 3.0 (AFI 3.0)
2. Affero GNU Public License
3. Adaptive Public License
4. Apache License, 2.0
5. Apple Public Source License
6. Artistic license 2.0
7. Attribution Assurance Licenses
8. New and Simplified BSD licenses
9. Boost Software License (BSL 1.0)
10. Computer Associates Trusted Open Source License 1.1
11. Common Development and Distribution License
12. Common Public Attribution License 1.0 (CPAL)
13. Common Public License 1.0
14. CUA Office Public License Version 1.0
15. EU DataGrid Software License
16. Eclipse Public License
17. Educational Community License, Version 2.0
18. Eiffel Forum License V2.0
19. Entessa Public License
20. European Union Public License (link to every language's version on their site)
21. Fair License
22. Frameworkx License
23. GNU General Public License (GPL)
24. GNU General Public License versión 3.0 (GPLv3)
25. GNU Library or "Lesser" General Public License (LGPL)
26. GNU Library or "Lesser" General Public License version 3.0 (LGPLv3)
27. Historical Permission Notice and Disclaimer
28. IBM Public License
29. IPA Font License
30. ISC License
31. Lucent Public License Version 1.02
32. Microsoft Public License (Ms PL)
33. Microsoft Reciprocal License (Ms-RL)
34. MIT license
35. Motosoto License
36. Mozilla Public License 1.1(MPL)
37. Multics License
38. NASA Open Source Agreement 1.3

39. NTP License
40. Naumen Public License
41. Nethack General Public License
42. Nokia Open Source License
43. Non-Profit Open Software License 3.0 (Non-Profit OSL 3.0)
44. OCLC Research Public License 2.0
45. Open Font License 1.1(OFL 1.1)
46. Open Group Test Suite License
47. Open Software License 3.0 (OSL 3.0)
48. PHP License
49. Python license (CNRI Python License)
50. Python Software Foundation License
51. Qt Public License (QPL)
52. RealNetworks Public Source License V1.0
53. Reciprocal Public License 1.5 (RPL 1 5)
54. Ricoh Source Code Public License
55. Simple Public License 2.0
56. Sleepycat License
57. Sun Public License
58. Sybase Open Watcom Public License 1.0
59. University of Illinois/NCSA Open Source License
60. Vovida Software License v 1.0
61. W3C License
62. wxWindows Library License
63. X.Net License
64. Zope Public License
65. Zlib/libpng license

