La empresa del software libre

Amadeu Albós Raya
Óscar David Sánchez Jiménez

PID_00201286

UOC
Universitat Oberta de Catalunya

© 2013, FUOC. Se garantiza permiso para copiar, distribuir y modificar este documento según los términos de la GNU Free Documentation License, Version 1.2 o cualquiera posterior publicada por la Free Software Foundation, sin secciones invariantes ni textos de cubierta delantera o trasera. Se dispone de una copia de la licencia en el apartado "GNU Free Documentation License" de este documento.

Índice

Introducción..... 5

1. Modelos de negocio..... 7
1.1. Desarrollo 8
1.2. Consultoría 11
1.3. Instalación e integración 12
1.4. Migración de sistemas 14
1.5. Administración y mantenimiento de sistemas 16
1.6. Soporte y formación 17

2. Plan de empresa..... 19
2.1. Resumen ejecutivo 21
2.2. Introducción 22
2.2.1. Misión y visión 22
2.3. Descripción del negocio 24
2.4. Organización de la producción 24
2.5. Organización interna y recursos humanos 26
2.6. Estudio de mercado 27
2.7. Plan de marketing 30
2.8. Análisis económico-finaciero 32
2.9. Forma legal 33
2.10. Gestión de riesgos 33
2.11. Resumen y evaluación 34
2.12. Plan de empresa y software libre 35

3. Producción de software libre..... 36
3.1. Creación y presentación del proyecto 37
3.2. Infraestructura 40
3.3. Organización de la comunidad 44
3.4. Desarrollo 47
3.5. *Releasing* y *packaging*..... 50
3.6. Elección de licencias 52

Resumen..... 54

Anexos..... 55

Introducción

Este módulo proporciona las bases para conocer los principales conceptos y características ligados al negocio empresarial del software libre.

Desde sus inicios, el software libre siempre ha estado presente a nivel de las tecnologías de la información, especialmente en el contexto empresarial. Su evolución a lo largo del tiempo se ha visto influida por los cambios estructurales que se han ido produciendo a nivel tecnológico, económico y social.

Ved también

Para conocer más aspectos históricos del software libre, consultad el segundo punto de los materiales de la asignatura de Introducción al software libre.

Con el paso del tiempo, se han puesto de manifiesto las diferentes filosofías en torno a la creación, producción y difusión del software. En términos generales, hay que destacar dos tendencias principales con características opuestas:

- Por un lado, la filosofía propietaria, que defiende la protección del software con el cierre y privatización del código fuente, así como la imposición de licencias con fuertes restricciones para su utilización.

- Por otra parte, la filosofía libre, que defiende la libertad del software y del código fuente con licencias que garantizan los derechos de los usuarios en la ejecución del programa, el estudio y la adaptación del código fuente y la redistribución y la publicación de las mejoras que se puedan introducir.

Estas dos filosofías han generado modelos de negocio empresarial opuestos en la ideología, el funcionamiento, la operativa y la economía:

- El modelo de software privativo normalmente establece un valor económico que hay que sufragar mediante el uso restringido de una copia en formato binario, hecho que imposibilita la adaptación, corrección o mejora del código fuente por parte de personas u organizaciones que no disponen de los derechos de autoría, o del acuerdo explícito de los que los poseen. Hay que destacar que muchas licencias propietarias impiden ceder los derechos de utilización a un tercero sin el previo acuerdo de los poseedores de los derechos de autoría.

- El modelo de software libre tiende a centrarse en el desarrollo y la adaptación de soluciones libres y cualitativas que respondan a las necesidades de usuarios y organizaciones, así como los servicios complementarios para su puesta en marcha y funcionamiento habitual. En este sentido, el modelo de negocio basado en el software libre permite las actuaciones que impide o restringe el modelo de software privativo.

La particular concepción filosófica del software libre no sólo influye directamente en el modelo de negocio y en la estrategia empresarial, sino también en la definición, la gestión, la organización y el funcionamiento de la empresa tecnológica. Aspectos como la madurez del software libre, la presencia de una comunidad mundial de colaboradores en proyectos de software libre y la viabilidad de los modelos de negocio reconfiguran el proyecto empresarial clásico de forma sustancial.

El primer apartado de este módulo se dedica a caracterizar a los principales modelos de negocio basados en el software libre, válidos y viables para ser llevados a la práctica como estrategia empresarial.

El segundo apartado se dedica íntegramente a la concepción del plan de negocio empresarial y se detallan los principales aspectos ligados a la creación, organización, producción y funcionamiento de la empresa del software libre.

El tercer apartado presenta la producción de software libre, con la caracterización de las principales particularidades en la creación, organización, comunicación y desarrollo del código fuente.

Finalmente, encontramos dos anexos en los cuales se presentan de forma breve y sistemática las principales licencias libres y estándares abiertos que tienen relación directa con el negocio del software libre.

1. Modelos de negocio

En este primer apartado se presentan los principales modelos de negocio basados en el software libre (a título orientativo y sin excluir otros planteamientos), así como las características y particularidades que lo diferencian de las tendencias de negocio basadas en software propietario.

En general, la diferencia más significativa entre el software libre y el privativo desde el punto de vista empresarial es la licencia. A grandes rasgos, una licencia es un modelo contractual en el cual el autor del producto (o aquél que posee los derechos de autoría) establece los derechos y deberes de los usuarios del producto y el escenario en el cual los pueden utilizar.

En este sentido, las licencias libres[1] se basan en cuatro principios básicos de libertad con respecto al software y su código fuente:

- La libertad de ejecutar el programa con cualquier propósito.
- La libertad de estudiar el código fuente y adaptarlo a las necesidades particulares, por lo cual es necesario el acceso al código fuente.
- La libertad de redistribuir copias del software.
- La libertad de mejorar el software y publicar las mejoras, por lo cual es necesario el acceso al código fuente.

(1)Existe una cierta controversia entre la Free Software Foundation (http://www.fsf.org) y la Open Source Initiative (http://www.opensource.org) a causa de las implicaciones de los términos *libre* y *abierto*.

Recurso web

Encontraréis la definición original de software libre en http://www.gnu.org/philosophy/free-sw.html.

Los principios básicos de libertad del software libre se contraponen al modelo privativo centrado en la explotación comercial de venta de licencias de utilización restringida del formato binario[2], aunque no se exige que el software libre se tenga que obtener de forma gratuita. Sin embargo, buena parte del software libre existente en la actualidad se puede obtener mediante descarga directa y gratuita desde el portal en Internet de la organización que lo gestiona.

(2)En inglés, *right-to-use licensing*.

Descargas directas y gratuitas

Unos ejemplos de descargas directas y gratuitas:

Debian GNU/Linux de http://www.debian.org/distrib/

FreeBSD de http://www.freebsd.org/where.html

KOffice de http://www.koffice.org/download/

OpenOffice.org de http://download.openoffice.org/

La apertura filosófica del software libre favorece los modelos de negocio centrados en el capital humano, los conocimientos, la personalización y la adecuación de los productos, así como la evolución constante del software. En este sentido, es importante destacar el papel que juega la comunidad de usuarios del software libre, que vela por la calidad y por la evolución de los productos libres, con un rendimiento difícilmente igualable en modelos de propiedad.

Con el tiempo, el modelo del software libre ha consolidado una oferta que abarca la mayoría de sectores con presencia de software privativo, y ofrece un mercado maduro, cualitativo y seguro sobre el cual basar una estrategia empresarial, tanto para el desarrollo de software como para los servicios complementarios.

Recursos web

En http://freshmeat.net/ y http://sourceforge.net/ encontraréis un amplio abanico de proyectos de software libre en las principales áreas de aplicación de la tecnología.

En cierta manera, la estrategia de negocio del software libre se fundamenta en los aspectos diferenciadores con respecto al modelo de propiedad, por ejemplo la ampliación de funcionalidades, la adaptación personalizada, el elevado y constante número de revisiones, las garantías de seguridad y calidad de funcionamiento del producto, así como todo un abanico de servicios complementarios para la puesta en marcha y el funcionamiento habitual.

En los siguientes apartados se presentan los principales modelos de negocio que se derivan de la filosofía conceptual del software libre: desarrollo, consultoría, instalación e integración, migración, mantenimiento y soporte y formación.

Los modelos de negocio que se presentan no deben entenderse como independientes, sino como complementarios. Es decir, puede convenir la combinación de dos o más modelos de negocio para dar respuesta a la estrategia empresarial.

1.1. Desarrollo

El modelo de negocio de desarrollo de software se basa en la producción total o parcial de un producto basado en software libre destinado a su comercialización, ya sea directamente o bien en el marco de un proyecto de implantación por cuenta ajena, como los presentados en la segunda unidad de la asignatura.

La definición de software libre no hace ninguna referencia con respecto a la estrategia de vender un producto libre a un precio por copia vendida, pero las características de las licencias libres convierten esta posibilidad en una opción secundaria, aunque sobradamente utilizada por algunas organizaciones.

Productos libres empaquetados

Algunas organizaciones deciden ofrecer sus productos libres empaquetados (caja, discos, manuales, documentación, etc.) a cambio del abono de un importe que, a pesar de ser inferior al precio de soluciones alternativas, es superior al precio de coste. Por ejemplo, se puede comprar la distribución Ubuntu en http://www.ubuntu.com/getubuntu/purchase.

En general, la producción de software libre responde principalmente a la venta de servicios complementarios de valor añadido para el cliente, que también sirven para sustentar la continuidad del software, por ejemplo la personalización y adecuación a un entorno determinado.

En los materiales de la asignatura de "Introducción al software libre" se ofrece una clasificación de las posibles alternativas del desarrollo de software libre, reproducida brevemente a continuación:

Ved también

Encontraréis más información y ejemplos de esta clasificación en el apartado 5.2 "Modelos de negocio basados en software libre" de los materiales de la asignatura de "Introducción al software libre".

- **Mejor conocimiento**. Se basa en el propósito de rentabilizar el conocimiento sobre uno o más productos libres, ofreciendo desarrollos a medida, modificaciones o adaptaciones (entre otros que se presentarán más adelante). La participación activa en la creación y desarrollo de los productos libres es el valor añadido de la empresa ante el cliente y la competencia.

- **Mejor conocimiento con limitaciones**. Se basa en el modelo anterior (mejor conocimiento) pero con una implementación mixta de licencias libres y de propiedad (o eventualmente patentes) para reducir la competencia. Este modelo puede resultar inviable si el producto libre se escinde en una rama soportado por la comunidad libre, de manera que podría desaparecer la ventaja competitiva.

- **Fuente de un producto libre**. Es similar al primer modelo (mejor conocimiento) con la diferencia que la empresa es la productora del código de forma casi íntegra. El cliente valora el posicionamiento y la ventaja competitiva con respecto a la competencia. Este modelo recibe el soporte de la comunidad libre.

- **Fuente de un producto con limitaciones**. Se basa en el modelo anterior (fuente de un producto libre) pero con una implementación orientada a reducir la competencia, por ejemplo iniciando la distribución bajo licencia privativa y liberándose más adelante, o haciendo que la distribución inicial se limite a los clientes de la empresa.

- **Licencias especiales**. Se basa en la producción de un mismo producto que se distribuye bajo diferentes licencias, libres y propietarias. El producto de propiedad se orienta a implementaciones especiales del producto, como la integración con otros productos propietarios.

- **Venta de marca**. Se basa en la distribución de productos libres con una imagen de marca empresarial, que da calidad y valor añadido desde el pun-

to de vista del cliente. Normalmente, estos productos se acompañan de numerosos servicios adicionales a los clientes.

La selección de la tipología de negocio asociada al desarrollo de software tiene que coordinarse principalmente con la estrategia empresarial y el mercado objetivo. En este sentido, una organización puede decidir usar una tipología personalizada para cada uno de los productos que pretenda introducir en el mercado, en función de la estrategia y el mercado objetivo particular que persiga en cada uno de ellos.

El modelo de negocio de desarrollo de software libre también exige la selección esmerada de las licencias del código fuente que manipula:

- La licencia del código fuente que modifica, si el producto final es una mejora de un producto que ya existía antes.

- La licencia del código fuente que enlaza, si el producto final requiere para su operativa implementar llamadas en funciones de bibliotecas externas.

- La licencia del código fuente que crea, es decir, cuando el código fuente es completamente nuevo.

- La licencia del código fuente del producto final, que considera el conglomerado de código fuente del producto final.

La importancia de determinar esmeradamente la licencia asociada a cada parte del código fuente que se manipula radica en la existencia de diferencias entre las diversas licencias libres. Es decir, aunque todas las licencias libres garantizan las cuatro libertades básicas, difieren en la política de licenciamiento de la redistribución de código modificado, que es precisamente el objeto del modelo de negocio de desarrollo de software libre.

En el anexo I de esta unidad presentamos brevemente las características más importantes de las principales licencias del software libre y exponemos la política de redistribución y las compatibilidades del enlace y de la obra derivada.

La selección y combinación adecuada de licencias influye directamente en la producción de software libre, y puede tener implicaciones legales si no se realiza esmeradamente. En el último apartado de esta unidad, dedicado a la producción de software libre, se profundizará en la selección de la licencia más adecuada a partir de los parámetros del producto.

Finalmente, hay que decir que el software libre promueve y utiliza especificaciones públicas, denominadas estándares abiertos, con el fin de favorecer la universalidad y la interoperabilidad de los formatos que manipulan. En el anexo II de esta unidad didáctica presentamos brevemente las principales características y ejemplos de los estándares abiertos.

1.2. Consultoría

El modelo de negocio de consultoría se basa en la producción de servicios profesionales complementarios al software libre para usuarios y organizaciones.

En cierta manera, este modelo de negocio se centra en el objetivo de proveer servicios tecnológicos profesionales externos de calidad a organizaciones que no asumen totalmente la creación, la gestión, el desarrollo y la evaluación de sus proyectos tecnológicos internos.

Ved también

En el apartado "Clasificación por ámbito" del primer módulo encontraréis más información de la tipología de proyectos productivos.

La diversidad de servicios profesionales que pueden ofrecer las consultorías es elevado, y dependerá de la estrategia y el entorno a actividad empresarial. Sin embargo, tienen una fuerte relación con las actividades de estudio, análisis, diseño y evaluación propias del proyecto de implantación de sistemas de software libre presentado en la segunda unidad de este módulo.

A continuación, presentamos una clasificación breve de los principales servicios que una consultoría puede ofrecer a sus clientes:

- **Gestión de proyectos**: se basa en el propósito de realizar las funciones de creación y gestión funcional del proyecto de implantación de software libre. Las tareas que desarrolla este servicio se relacionan con el ciclo de vida del proyecto, la gestión de los equipos de profesionales involucrados en cada etapa del proyecto, el control del progreso efectivo del proyecto y, en general, todas aquellas tareas de coordinación, información, gerencia y seguimiento del proyecto.

Ved también

En el apartado "Tipologia de proyectos" del primer módulo y en el apartado "Ciclo de vida" del segundo módulo encontraréis más información sobre la gestión y el ciclo de vida de los proyecto de software libre.

- **Ejecución de etapas de análisis y diseño del proyecto**: se basa en el propósito de realizar una o más etapas de análisis y diseño del proyecto de implantación de un sistema en software libre. Las tareas que desarrolla este servicio se relacionan con los encargos de la contratación, por ejemplo el estudio del sistema actual, el estudio de requerimientos del nuevo sistema, el análisis de soluciones en software libre y/o el diseño de un nuevo sistema, de acuerdo con las etapas presentadas en la segunda unidad del módulo.

Ved también

En los apartados "Estudio de la situación actual", "Estudio de los requisitos de la implantación", "Análisis de las soluciones en software libre" y "Desarrollo" del segundo módulo encontraréis más información sobre el estudio del sistema actual, el estudio de requerimientos del sistema, el análisis de las soluciones en software libre y el diseño del sistema, respectivamente.

- **Evaluación y auditoría**: se basa en el propósito de realizar valoraciones tecnológicas profesionales de una o más características de sistemas en fun-

cionamiento. Las tareas que desarrolla este servicio pueden estar enmarcadas dentro de un proyecto de implantación de sistemas, como sería el caso de la ejecución de etapas visto anteriormente, pero también se pueden desarrollar de forma independiente y aislada. La evaluación o la auditoría se centra en valorar uno o más aspectos del sistema, como la seguridad, el rendimiento, la eficiencia o la eficacia (entre otros), y se puede realizar antes y/o después de la implantación del sistema.

- **Asesoría:** se basa en el propósito de dar soporte, ayuda y consejo profesional a la toma de decisiones tecnológicas en la organización. Las tareas que desarrolla pueden tener lugar antes del inicio de cualquier proyecto de implantación o bien en las fases de estudio, análisis y diseño, pero se enmarcan en el soporte profesional a decisiones estratégicas tecnológicas para el futuro de la organización.

La lista presentada no se tiene que considerar ni exhaustiva ni excluyente, ya que el modelo de negocio puede estructurar dos o más servicios para responder a la estrategia empresarial. Adicionalmente, el modelo de negocio de consultoría puede combinarse con otros modelos de negocio para ofrecer al cliente un servicio tecnológico integral.

En este sentido, el mejor conocimiento de las plataformas tecnológicas implantadas (o que hay que implantar), así como la excelencia para el análisis y extracción de información y conclusiones, o el alcance y la complejidad del proyecto, son características decisorias a fin de que la organización decida externalizar la gestión o las etapas de la implantación.

Normalmente, los trabajos de consultoría se formalizan con contratos abiertos o cerrados. En los contratos abiertos la relación se origina a partir del encargo de un servicio determinado y, en función del resultado del servicio, el contrato se amplía con el encargo de nuevos servicios. Por ejemplo, la ejecución de una o más etapas del proyecto puede estar sujeta a contratos abiertos.

En cambio, en los contratos cerrados se adjudica el cumplimiento de un determinado objetivo, función, resultado o trabajo, sin posibilidad directa de ampliar la contratación en el mismo escenario. Por ejemplo, la auditoría independiente de un sistema puede estar sujeta a contratos cerrados, dadas las eventuales características temporales de puntualidad y aislamiento.

1.3. Instalación e integración

El modelo de negocio de instalación e integración se basa en la implantación directa de sistemas en software libre para usuarios y organizaciones, normalmente enmarcados en proyectos de software libre.

En cierta manera, este modelo de negocio considera el software libre como objeto para la producción de servicios, más que como producto en sí mismo. De esta forma, se fundamenta un mercado en el cual se resaltan los beneficios para el cliente:

- La organización no tiene que pagar licencias en productos de software libre que se distribuyen de forma gratuita y, por consiguiente, puede reducir el coste de implantación tecnológica.

- La organización no tiene que recurrir a la piratería de productos y, por consiguiente, no incumplirá la legalidad vigente.

- La organización puede adaptar directamente las soluciones libres y, por consiguiente, podrá disminuir el coste de implantación de sistemas especializados.

- La organización puede adoptar paquetes integrados de implantación directa y, por consiguiente, reducirá el riesgo de implantación tecnológica.

A continuación presentamos una lista breve de los principales servicios que puede ofrecer este modelo de negocio, aunque la lista no es exhaustiva ni excluyente de otros servicios:

- **Configuración**: se trata de realizar las tareas de configuración y ajuste[3] de un sistema ya implantado con el objetivo de formalizar la configuración inicial, mejorar su rendimiento o ajustarlo a nuevos propósitos no considerados inicialmente. En cualquier caso, los ajustes que se introducen no modifican el código fuente de la aplicación, únicamente la configuración de aquellos componentes que permitan el ajuste a las características particulares de la instalación.

- **Pruebas**: se trata de proporcionar un banco de pruebas de sistemas, aplicaciones o soluciones en software libre bajo una perspectiva determinada. Puede tratarse de un análisis comparativo de soluciones libres, las pruebas de un nuevo diseño de sistema, o el test de software de implantación directa sobre hardware específico, ya sea en el marco de un proyecto de implantación, o bien como pruebas independientes y aisladas.

- **Integración**: se trata de realizar y/o comprobar la integración entre dos o más soluciones de software libre, con el objetivo de proporcionar un paquete único que resuelva una función operativa concreta[4]. Normalmente, esta integración se puede resolver con la configuración apropiada de cada elemento y, eventualmente, con algún componente adicional que permita la integración con más eficiencia.

(3)En inglés, *set-up* o *tune-up*.

Ved también

En los apartados "Análisis de las soluciones en software libre", "Desarrollo" e "Implantación y migración" del segundo módulo encontraréis más información sobre el análisis de las soluciones en software libre, el diseño del sistema y la implantación y migración del sistema, respectivamente.

(4)Por ejemplo, LAMP (*Linux, Apache, MySQL, PHP*) es un paquete integrado de software con diferentes objetivos particulares, pero que conjuntamente soluciona una problemática concreta de forma eficiente y eficaz.

- **Instalación**: se trata de realizar la instalación masiva de software de implantación directa en máquinas clientes o servidores[5]. Este servicio puede necesitar la configuración y el ajuste tanto del software libre que se pretende instalar como del hardware receptor de la instalación. También puede ser necesaria la integración de las diferentes soluciones que se pretenden instalar. En los casos en los cuales se tenga que implantar el mismo software en un conjunto de ordenadores con idéntico hardware, puede resultar útil usar software de clonado y distribución de imágenes preconfiguradas para ahorrar tiempo y coste, además de mejorar la eficiencia y eficacia del proceso.

(5)En inglés, sería similar al término *installfest*, pero aplicado al negocio estructurado.

- **Distribución**: se trata de redistribuir software libre a los clientes, ya sea en el format original[6] o bien en configuraciones personalizadas en relación con el ámbito de negocio empresarial; por ejemplo la integración de herramientas, la orientación operativa a cliente, a servidor o a estación de trabajo[7], entre otros. La redistribución de software integrado está sujeta a las licencias de las soluciones particulares. En el anexo I se presentan las principales licencias de software libre y su compatibilidad.

(6)En http://freshmeat.net/ y http://sourceforge.net/ encontraréis un amplio abanico de proyectos de software libre en las principales áreas de aplicación de la tecnología.

(7)En inglés, *workstation*.

Tal como se desprende de la clasificación anterior, estos servicios tienen una fuerte relación con las etapas de análisis de las soluciones en software libre y de implantación y migración del proyecto de implantación presentado en la segunda unidad. En este sentido, el mejor conocimiento de las plataformas tecnológicas, así como la excelencia y eficiencia de los servicios, o el alcance y la complejidad del proyecto, son características decisorias a fin de que la organización decida externalizar la instalación y la integración del sistema.

1.4. Migración de sistemas

El modelo de negocio de migración de sistemas se basa en el hecho de trasladar la operativa de funcionamiento entre el sistema implantado y el nuevo sistema que hay que implantar.

La migración es un proceso complejo, que hay que ejecutar con la precisión y el rigor necesarios de acuerdo con la importancia que suponen los datos y configuraciones como capital para la organización.

La diversidad de situaciones con que se pueden encontrar las empresas que se dedican a la migración de sistemas son fruto de la combinación entre las plataformas de origen y de destino de la migración. El conocimiento profundo de las plataformas y la experiencia en la migración fundamentan el valor añadido para el cliente.

A continuación presentamos una lista breve de los principales servicios que ofrece este modelo de negocio, aunque la lista no es exhaustiva ni excluyente de otros servicios:

- Servicios de sistema de ficheros, tanto de servidor como de clientes.
- Servicios de impresión, tanto entre clientes como entre servidores.
- Servicios de directorio y de autentificación centralizada.
- Servicios de red, especialmente de protocolos de gestión automatizada de control de la red, de las comunicaciones y de los clientes.
- Gestión y administración del sistema, tanto la gestión de la red como del software.
- Servicios web, tanto con plataformas estáticas como dinámicas.
- Bases de datos, tanto la migración de los datos como la verificación de los accesos.
- Entornos de escritorio y aplicaciones de ofimática, tanto las aplicaciones como los datos de usuario.
- Aplicaciones corporativas, tanto las aplicaciones que se pueden ejecutar directamente, como las que exigen ajustes o virtualizaciones.

Ved también

En el apartado "Implantación y migración" del segundo módulo encontraréis sobradamente detalladas las características de la migración de sistemas. Concretamente, en el apartado "Migración de los servicios de un sistema" encontraréis más información de los servicios que aquí presentamos.

La complejidad y el alcance de la migración, la capacidad de realizar el proceso de forma esmerada, eficiente, eficaz y con el menor tiempo posible, así como el mejor conocimiento de las plataformas tecnológicas de origen y de destino de la migración son características decisorias a fin de que la organización decida externalizar la migración del sistema.

Tal como se puede concluir de la clasificación de los servicios que hemos presentado, el proceso de migración puede necesitar otros servicios, como la instalación, la configuración, la integración o las pruebas para asegurar la consecución de los objetivos.

El software libre utiliza y fomenta los estándares abiertos para el intercambio interoperable de los datos y es especialmente relevante su papel en la migración de los sistemas. Por ejemplo, partir de un sistema que no almacene los datos en estándares abiertos podría complicar la migración a causa de la conversión de los formatos, especialmente si el original es privativo. En el anexo II de este módulo didáctico presentamos los estándares abiertos, su definición, los organismos que los impulsan y algunos ejemplos.

1.5. Administración y mantenimiento de sistemas

El modelo de negocio de administración y mantenimiento de sistemas se basa en realizar las tareas de gestión y seguimiento de un sistema ya implantado y en funcionamiento.

El principal objetivo de los servicios que puede ofrecer este modelo de negocio es mantener funcional todo el sistema a lo largo del tiempo, ajustando la configuración a los cambios que se produzcan, solucionando problemáticas que puedan surgir y reparando las averías que impidan el funcionamiento habitual de la organización.

A continuación presentamos una lista breve de los principales servicios que ofrece este modelo de negocio, aunque la lista no es exhaustiva ni excluyente de otros servicios:

- **Administración**: consiste en proporcionar la gestión principal del sistema, el ajuste necesario con el paso del tiempo, la supervisión del funcionamiento, la implantación de nuevas funcionalidades y la evolución del sistema. Muchas de las tareas de administración del sistema se pueden ejecutar de forma remota[8].

- **Mantenimiento y evolución**: consiste en proporcionar supervisión, seguimiento y corrección de las incidencias que se puedan originar en el sistema y que impidan su funcionamiento, así como el control y evolución de la obsolescencia de los componentes. Por ejemplo, las averías y la desconfiguración de hardware o de software, el control y actualización de versiones de software, o el plan de evolución de hardware y software.

- **Seguridad**: consiste en proporcionar la gestión de la seguridad del sistema, controlando los riesgos, manteniendo políticas de prevención, contingencia, diagnóstico y corrección de fallos. Por ejemplo, copias de seguridad o el control y mantenimiento de claves y certificados.

Dadas las características y particularidades de estos servicios, muchas organizaciones deciden mantener trabajadores en plantilla que realicen estas tareas, pero algunas organizaciones pequeñas o medias no tienen bastante capacidad para asumir esta figura.

(8)Por ejemplo, el uso conjunto de `ssh`, `cron` y `at`.

También hay que tener en cuenta que la externalización de algunos servicios, como servidores de intranet y de extranet, pueden motivar la contratación de servicios externos de administración y mantenimiento. Normalmente, estos servicios funcionan con contratos con cuota fija mensual o anual para una determinada cobertura de servicios.

Intranet y extranet

Los servicios web, de intranet o de extranet son fácilmente externalizables gracias a la proliferación de los hoteles de datos (*Data Hotels*) o centros de datos (*Data Centers*).

1.6. Soporte y formación

El modelo de negocio de soporte y formación consiste en proporcionar ayuda técnica profesional para la educación y aprendizaje tecnológico de usuarios y la resolución de incidencias y problemas relacionados con la explotación del sistema.

La implantación de sistemas en software libre puede necesitar medidas de soporte y formación a los usuarios durante los primeros momentos de utilización, especialmente si el sistema anterior estaba basado en software privativo. Tal como se ha visto en el segundo módulo de la asignatura, el proyecto de implantación tiene que tener en cuenta la necesidad de actuar en el aprendizaje de los usuarios para favorecer la gestión del cambio, el cual, junto con sus particularidades, hace que acabe siendo un servicio fácilmente externalizable a empresas especializadas en este sector.

A continuación presentamos una lista breve de los principales servicios que ofrece este modelo de negocio, aunque la lista no es exhaustiva ni excluyente de otros servicios:

- **Formación**: se propone proporcionar educación y aprendizaje sobre herramientas de software libre a usuarios finales, por ejemplo sistemas operativos o herramientas de ofimática. Este servicio también puede incluir la formación de software especializado resultante del desarrollo producido en el proyecto de implantación de sistemas, por lo que puede resultar conveniente para la gestión del cambio coordinar los esfuerzos con el equipo de implantación.

Ved también

En el apartado "Formación, comunicación y soporte al usuario" del segundo módulo encontraréis más información sobre la formación y el soporte al usuario.

- **Soporte**: se propone proporcionar asistencia técnica a los usuarios ante problemas cotidianos. Muchos de estos servicios se implementan en centros telefónicos de ayuda[9], aunque también puede resultar útil ofrecer buzones de correo electrónico para la resolución de incidencias o mensajería instantánea[10] con un profesional. También puede resultar conveniente combinar los esfuerzos con el equipo del proyecto de implantación de sistemas con el fin de resolver posibles errores del software implantado.

[9]En inglés, *call centers*.

[10]En inglés, *instant messaging* o *chat*.

En general, este modelo de negocio pide recursos humanos, tecnológicos y materiales adecuados a los objetivos de la formación:

- Recursos humanos con conocimientos profundos de la temática y con capacidad para transmitir los conocimientos y resolver problemas.

- Recursos tecnológicos adecuados a la formación y el soporte, por ejemplo plataformas tecnológicas de ayuda a la educación o centros telefónicos de asistencia técnica.

- Recursos materiales adecuados a la formación, como documentación y manuales específicos de software libre[11] bajo licencias libres.

La calidad de estos parámetros es fundamental a fin de que la organización decida externalizar los servicios de formación y soporte. Normalmente, los servicios de formación se contratan por cursos con estructura acordada previamente, mientras que los servicios de soporte se contratan por cuotas mensuales o anuales, con acuerdo previo de los servicios cubiertos.

Web complementaria

Por ejemplo, Moddle es una plataforma de educación a distancia basada en software libre. http://moodle.org/.

(11)El proyecto SELF proporciona una plataforma europea para crear y compartir materiales educativos relacionados con el software libre y los estándares abiertos (http://selfproject.eu/).

Ved también

En el apartado 10.2 "Licencias de otros recursos libres" del material de la asignatura de Introducción al software libre encontraréis dos licencias de documentación, materiales y obras literarias sobradamente utilizadas en la actualidad.

2. Plan de empresa

Un plan de empresa o de negocio es un instrumento que identifica, describe y analiza una oportunidad de negocio, examina su viabilidad y desarrolla los procedimientos y estrategias para crear la empresa que explote la oportunidad de negocio.

Aclaración

En este apartado se utilizará preferentemente el término plan de empresa, ya que el objetivo del apartado es presentar los elementos necesarios para la creación de una empresa de software libre, como la ilustrada en el caso Cometa Technologies, que se verá en el segundo apartado.

De acuerdo con esta definición, los objetivos del plan de empresa son los siguientes:

- Realizar un estudio de mercado que posicione el proyecto empresarial y determine su viabilidad técnica, económica y financiera
- Desarrollar las medidas necesarias para conseguir los objetivos fijados en el mismo plan de empresa
- Realizar un seguimiento de la evolución de la empresa y evaluar las desviaciones respecto al plan de empresa inicial
- Servir como tarjeta de presentación del proyecto y de los emprendedores a fin de obtener financiación y apoyo de terceros

Si bien los tres primeros objetivos son principalmente de uso interno, el último es de uso externo y visible por personas ajenas al proyecto, al menos en principio. Durante la preparación de un plan de empresa hay que tener en cuenta siempre esta doble finalidad: servir como plan de proyecto y, a la vez, como presentación del proyecto.

Por supuesto, hay que evitar la tentación de omitir los riesgos o aspectos negativos del proyecto a fin de hacerlo más atractivo a los inversores. De hecho, la falta de estos elementos podría volverse en contra del propio proyecto empresarial, ya que estaría basado sobre supuestos falsos. Por ello, la veracidad de los aspectos técnicos y económicos es uno de los requisitos básicos a la hora de redactar un plan de empresa.

Todo plan de empresa debe responder a una serie de preguntas sobre el proyecto que se desea poner en marcha: ¿quién?, ¿qué?, ¿por qué?, ¿dónde?, ¿cuándo? y ¿cuánto?

- **¿Quién?**
  El nombre de la empresa, la marca de los productos o servicios ofrecidos, los nombres y trayectoria de los promotores.

- **¿Qué?**
  La descripción de los productos o servicios ofrecidos, los mercados a los que se dirigen y la cuota de mercado que se fija como objetivo, entre otros.

- **¿Por qué?**
  En general, todo plan de empresa busca obtener y maximizar los beneficios económicos. Sin embargo, esto no es incompatible con otros objetivos como la mejora de la calidad de vida de la sociedad o la creación de empleo.

- **¿Dónde?**
  La zona geográfica en la que se van a comercializar los productos o servicios, por ejemplo regional, nacional o internacional. Los canales de distribución que se van a utilizar, incluyendo los posibles acuerdos con otras empresas que permitan acceder a otras regiones.

- **¿Cuándo?**
  El inicio previsto de la actividad empresarial y su planificación posterior, incluyendo condiciones o limitaciones temporales que puedan afectar a la empresa, como los trámites de obtención de licencias, el tiempo de producción, la obsolescencia de determinadas tecnologías o la estacionalidad.

- **¿Cuánto?**
  La inversión inicial necesaria para poner en marcha el proyecto empresarial, la facturación mínima necesaria y la deseada, el umbral de beneficios y pérdidas, la reinversión de los beneficios y la repartición de dividendos, entre otros.

Estas cuestiones se traducen en los siguientes aspectos que se encuentran en casi todo plan de empresa:

- Resumen ejecutivo

- Introducción

- Descripción del negocio

- Organización de la producción

- Organización interna y recursos humanos
- Estudio de mercado
- Plan de marketing
- Análisis económico-finaciero
- Forma legal
- Gestión de riesgos
- Resumen y evaluación

Dependiendo de la naturaleza de la empresa o negocio cada uno de estos aspectos tendrá mayor o menor importancia en el plan de empresa y podrá organizarse de diferentes maneras.

En los apartados que veremos a continuación se revisará cada uno de estos aspectos y se estudiará su relación con los modelos de negocio basados en software libre vistos en el apartado anterior.

2.1. Resumen ejecutivo

El resumen o sumario ejecutivo es una nota breve[12] que aparece al principio del plan de empresa y que resume los elementos principales del documento. De esta manera, permite que por ejemplo potenciales inversores puedan hacerse una idea completa del plan de empresa, sin necesidad de entrar en los detalles de cada uno de los apartados.

(12)En cualquier caso, su extensión no debería superar las tres páginas.

El resumen ejecutivo debe repasar prácticamente todos los aspectos del plan de empresa, a saber:

- Descripción del modelo de negocio, prestando especial atención a la cadena de valor y la fuente de ingresos.
- Breve descripción del los promotores del proyecto, su formación, sus conocimientos y habilidades, su trayectoria profesional y su dedicación al nuevo proyecto.
- Descripción concisa del mercado, incluyendo tamaño, clientes, potencial de crecimiento y barreras.

- Análisis de las áreas funcionales del proyecto: producción, calidad y organización de los recursos humanos.

- Resumen del análisis financiero del proyecto y de la inversión necesaria para su puesta en marcha.

- Resumen de los riesgos asociados al proyecto y los planes para prevenirlos y remediarlos.

Resulta obvio que el resumen ejecutivo debe destacar los puntos fuertes del plan de empresa, especialmente en lo que se refiere al modelo de negocio que se desea desarrollar, la estrategia que se va emplear para ello y el equipo promotor.

Se recomienda redactarlo una vez el plan de empresa está terminado y hacerlo desde cero, es decir, evitando reutilizar textos ya escritos.

2.2. Introducción

A continuación del resumen ejecutivo y el índice, el primer elemento del plan de empresa debe ser una introducción que presente el nombre de la futura empresa[13] y el equipo promotor, así como el resto de profesionales involucrados en la redacción del plan de empresa.

(13)En el caso de que el plan describa un nuevo producto o servicio de una empresa ya constituida, es conveniente presentar un resumen de su actividad, su evolución histórica, su tamaño, etc.

La presentación del equipo promotor debe incluir, como hemos visto, el historial profesional de cada uno de sus miembros y los conocimientos que aportan al proyecto empresarial. Es bastante frecuente que una parte del equipo promotor tenga un perfil especializado en gestión empresarial, pero que también cuente con especialistas en determinadas áreas tecnológicas, como en el caso de las empresas que trabajan con software libre.

Por último, la introducción debe proporcionar una breve descripción de los diferentes apartados del plan de empresa que se desarrollarán posteriormente.

2.2.1. Misión y visión

La introducción es un buen lugar para presentar la misión y la visión de la nueva empresa, de manera que el lector pueda comprobar como estos dos conceptos se desarrollan en el plan de empresa.

La misión y la visión de una organización permiten definir de una manera concisa sus principales características y objetivos y las estrategias para llevarlos a cabo.

La misión consiste en una frase concisa que justifica la existencia de la organización, es decir el propósito básico hacia el que apuntan sus actividades, y los valores que guían las actividades de sus empleados. La misión está fuertemente vinculada a los valores internos de la organización, y describe en buena medida cómo competir y generar valor al cliente.

La visión es también una frase concisa que describe las metas a medio y largo plazo de la organización. La visión está orientada al mercado y debe expresar de una manera colorista y visionaria cómo quiere la organización ser percibida por el mundo.

Las principales diferencias entre misión y visión se resumen en lo siguiente:

- La misión describe los aspectos internos de la organización y su funcionamiento, mientras que la visión describe los aspectos externos.

- La misión tiene su horizonte en el corto y medio plazo, poniendo especial énfasis en los aspectos que se deben poner en práctica inmediatamente en la organización, mientras que la visión se fija a medio y largo plazo y da las líneas generales de la evolución de la organización en el futuro.

Corcaribe Tecnología y eZ Systems

La empresa Corcaribe Tecnología, especializada en productos y servicios basados en software libre, se fija la siguiente misión:

"Corcaribe Tecnología provee soluciones tecnológicas que generan valor agregado bajo un modelo de negocio que permite ofrecer a sus clientes el mejor coste por los resultados entregados, produciendo auténticos beneficios tangibles e intangibles a sus miembros y colaboradores."

Y la siguiente visión:

"Convertirse en referencia latinoamericana de continuo éxito en la implementación de soluciones tecnológicas integrales aplicando preceptos y valores del conocimiento libre dentro de un modelo de desarrollo sustentable."

Del mismo modo, eZ Systems, que ofrece software libre de gestión de contenidos se propone la siguiente misión:

"Ser la primera plataforma de gestión de contenidos en 2012."

Y la siguiente visión:

"Ayudar a las empresas a gestionar, publicar y compartir la información."

La definición de una misión y una visión no es imprescindible en un plan de empresa, pero puede ayudar a sintetizar los objetivos a corto, medio y largo plazo del proyecto empresarial y a transmitirlos eficazmente a los potenciales inversores.

2.3. Descripción del negocio

Es recomendable comenzar este apartado con una descripción de la empresa que se desea crear y con una breve presentación de los promotores del proyecto, aunque ya se haya realizado previamente en la introducción.

El objetivo esencial de este apartado es describir los productos o servicios para los cuales se está realizando el plan de empresa o de negocio, así como el modelo de negocio bajo el que van a ofrecerse, tal como se ha visto en el apartado anterior.

Se debe prestar especial atención a explicar las particularidades de los modelos de negocio basados en software libre y tener siempre en cuenta que el lector del plan de empresa no tiene por qué estar familiarizado con ellos; por ejemplo, en los aspectos relacionados con la protección de la propiedad intelectual y los derechos sobre los productos o servicios.

Del mismo modo, es conveniente explicar las necesidades que los productos o servicios van a cubrir y las principales diferencias respecto a la oferta ya existente, a fin de mostrar que el proyecto empresarial está bien posicionado en el mercado.

Finalmente, es necesario presentar la capacidad de producción y prestación de servicios, lo que servirá de introducción al próximo apartado dedicado a la organización de la producción.

2.4. Organización de la producción

Dentro del plan de empresa, el apartado dedicado a la organización de la producción aporta una descripción de las tareas técnicas de la futura empresa.

Hasta ahora se ha contemplado la posibilidad de que el plan de empresa o de negocio describa la comercialización de un nuevo producto o servicio, indistintamente. Según se trate de uno u otro, este apartado del plan de empresa tendrá una de las siguientes formas:

- En el caso de que la actividad de la empresa esté basada en el desarrollo, producción y posterior comercialización de un producto, se describirán en detalle las fases de desarrollo y producción.

- En el caso de que la empresa se dedique a la prestación de un servicio sin ningún proceso productivo, se describirán en detalle los procedimientos para la prestación del servicio y las necesidades técnicas.

Por supuesto, estos dos casos no son excluyentes y en un plan de empresa se pueden dar ambas. Por ejemplo, una empresa especializada en migración de sistemas a software libre que también ofrece servicios de formación en tecnologías basadas en software libre a usuarios y a personal técnico.

En general, la actividad empresarial que implica las fases de investigación, desarrollo y producción presenta una complejidad mucho mayor, y también mayores riesgos:

- **Fase de investigación y desarrollo**. La descripción de la fase de investigación y desarrollo debe prestar especial atención a la estimación de la duración de la fase de investigación y desarrollo y de las necesidades de inversión en recursos materiales y humanos.
  Especialmente en sectores de alta tecnología, como es el caso del software libre, el plan de empresa debe evaluar la capacitación de los recursos humanos y el *know-how* necesarios para el éxito de las tareas de investigación y desarrollo. Por otra parte, este apartado debe describir en detalle la distribución de funciones y responsabilidades, los riesgos inherentes a toda actividad de investigación y desarrollo, las potenciales sinergias entre proyectos, el proceso de innovación y mejora continua de los productos y cómo este proceso va a integrarse en el proceso de producción. D'altra banda, aquest apartat ha de descriure en detall la distribució de funcions i responsabilitats, els riscos inherents a tota activitat d'investigació i desenvolupament, les potencials sinergies entre projectes, el procés d'innovació i millora continuada dels productes, i la manera com s'integrarà aquest procés en el procés de producció.

- **Fase de producción**. La descripción del proceso productivo debe describir en primer lugar el ciclo operativo[14], la localización de las instalaciones de producción, su coste y su accesibilidad. En segundo lugar, es necesario describir los locales, edificios y equipos necesarios para la producción o prestación de los servicios. Para cada uno de éstos se deben presentar las modalidades de financiación y adquisición[15], sus características, su disponibilidad, su duración y su amortización anual.

(14) Esto incluye la capacidad productiva en número de unidades y la producción esperada, así como el personal y el número de horas o turnos necesarios para la producción.

(15) Igualmente, se puede presentar los planes de expansión de las instalaciones y de adquisición de nuevos equipos.

Se debe prestar especial atención a la gestión de la calidad, con la descripción de los estándares y certificaciones de calidad que se van a aplicar tanto a los procesos como a los resultados del proceso productivo.

Finalmente, se debe dar una visión estratégica del proceso productivo, por ejemplo en el caso de que se subcontrate la producción de algunos componentes o una parte del proceso productivo[16].

(16)Por ejemplo, un editor de software libre podría subcontratar la producción del soporte de distribución y del embalaje de sus programas.

De nuevo, la descripción del proceso productivo de software libre presenta una serie de diferencias respecto al desarrollo de software propietario, que deben ser explicadas adecuadamente en el plan de empresa, aún más cuando el lector puede no estar familiarizado con el software libre. Igualmente, es conveniente incidir en la calidad añadida que supone el software libre respecto al software propietario.

Ved también

En el apartado "Producción de software libre" de la tercera unidad se estudian en detalle las particularidades del proceso de producción de software libre.

En cualquier caso hay que recordar que siempre se deben presentar las ventajas y desventajas de las distintas alternativas y justificar cada una de las decisiones.

2.5. Organización interna y recursos humanos

Este apartado del plan de empresa concreta la organización del equipo de trabajo necesario para el desarrollo del proyecto empresarial y los perfiles necesarios.

En primer lugar, se debe incluir una descripción de las funciones y puestos de dirección clave, junto a los perfiles necesarios e incluso el nombre y currículo[17] de las personas que vayan a ocupar estos puestos en el caso de que ya estén decididas. En segundo lugar, se deben describir las diferentes categorías profesionales necesarias en la empresa, sus responsabilidades, las principales tareas que llevarán a cabo y la modalidad de contratación, entre otros. Es conveniente indicar la remuneración correspondiente a cada tipo de trabajadores ya ocupen cargos directivos o no.

(17)Incluye su experiencia profesional, su especialización y los principales logros en su carrera. Este tipo de información cumple una doble función: por un lado refuerza la confianza de los inversores potenciales y, por otro lado, permite descubrir las fortalezas y debilidades del equipo gestor.

La organización interna de la empresa se puede representar fácilmente mediante un organigrama, por departamentos y áreas de actividad y con las personas específicas, si existieran, en los puestos de dirección.

Este apartado debe concluir con la descripción de la política general de la empresa en el área de recursos humanos y decidir si es necesaria la creación de un departamento específico de recursos humanos o la gestión de éstos puede realizarse de forma distribuida en cada departamento.

La necesidad y la disponibilidad de personal cualificado en una determinada área y a un coste apropiado puede suponer en ocasiones una barrera de entrada importante, como puede ser el caso de especialistas en software libre[18].

(18)Por ejemplo, en el momento de la edición de estos materiales, si bien el software libre es ya conocido y las tecnologías y soluciones basadas en él bastante populares, es difícil encontrar profesionales que hayan participado activamente en proyectos de software libre, ya sea como empleados en una empresa o por propia iniciativa.

Además, una empresa que base su modelo de negocio en software libre puede precisar de puestos y responsabilidades adaptados a sus particularidades. Por ejemplo, junto a los puestos clásicos de director técnico o director comercial, se pueden encontrar roles como director de comunidad, responsable de la gestión de las relaciones con los desarrolladores y usuarios de software libre o director de proyectos en cooperación, responsable de la gestión y coordinación de proyectos que se lleven a cabo en colaboración con otras empresas, centros de investigación o universidades.

2.6. Estudio de mercado

El estudio de mercado es una parte fundamental de un plan de empresa, y por tanto una de las claves de su éxito.

Un buen estudio de mercado va a permitir evaluar correctamente la viabilidad técnica y económica del proyecto empresarial y a identificar los potenciales clientes y competidores, de manera que se pueda definir la estrategia adecuada para vender los productos o servicios objeto del plan de empresa.

En la elaboración de un plan de empresa es conveniente realizar el estudio de mercado en primer lugar, o al menos en una primera aproximación, ya que sus resultados pueden afectar a diferentes partes del plan de empresa.

Así, el análisis de mercado debe proporcionar información sobre los aspectos siguientes:

- **Situación actual del mercado**. En primer lugar, es necesario segmentar el mercado de acuerdo a las características más relevantes en el plan de empresa y determinar su volumen, así como su evolución histórica. Igualmente, hay que determinar el proceso de decisión en el mercado y el comportamiento de los clientes, en particular su reacción ante la introducción de nuevos productos o servicios.
  En segundo lugar, hay que evaluar las posibles necesidades que podría generar la introducción de los productos o servicios propuestos en el plan de empresa. Esto depende en buena medida de si el producto o servicio aporta algo nuevo y de la capacidad de influir en los hábitos de los clientes.

- **Previsiones sobre el crecimiento del mercado**. Una vez se conoce el estado actual del mercado hay que ser capaz de realizar previsiones sobre su evolución futura. ¿Se trata de un mercado en crecimiento, estable o en

decadencia? ¿Cuál es el nivel de fragmentación del mercado? ¿Se está produciendo un proceso de concentración?

De nuevo, hay que tener en cuenta la influencia que los nuevos productos o servicios podrían tener en el mercado. Por ejemplo, la introducción de soluciones basadas en software libre puede efectivamente cambiar el mercado, al propiciar la formación de un nuevo sector especializado en software libre.

- **Identificación y clasificación de los clientes**. Uno de los objetivos fundamentales de un análisis de mercado es descubrir quiénes serán los clientes potenciales de los productos y servicios propuestos. La labor de clasificar los diferentes tipos de clientes de acuerdo a ciertas características comunes es también muy importante, ya que permite definir diferentes estrategias para cada uno de ellos. Por ejemplo, una empresa que ofrezca la implantación de sistemas de software libre para empresas se presentará de manera distinta a sus clientes según éste sea una empresa familiar o una gran corporación. Hay que tener en cuenta también que un mismo producto o servicio puede ofrecerse a clientes a priori diferentes. La flexibilidad e interoperabilidad del software libre favorece esto último.

  Por otra parte, se debe evaluar la recepción del producto o servicio por parte de cada tipo de cliente. Siguiendo con el ejemplo anterior de una empresa especializada en la implantación de sistemas de software libre, una gran empresa que cuente con personal técnico dedicado puede ser más reticente a la adopción del software libre, en parte por el miedo al cambio, mientras que una empresa familiar puede ser más receptiva.

  Finalmente, en el caso de que la futura empresa cuente ya con una cartera de clientes, o de que existan clientes que hayan mostrado su interés en sus productos o servicios, es conveniente recoger esto en el estudio de mercado.

- **Análisis de la competencia y de sus productos**. El estudio de mercado debe dar cuenta de los competidores de la futura empresa, así como identificar tanto sus fortalezas y debilidades como las de los productos o servicios que ofrecen.

  Se debe proporcionar información sobre las características de sus productos y servicios, incluyendo su precio y calidad, así como su cuota de mercado y su estrategia comercial. Es muy importante identificar los líderes en el mercado de cada uno de los productos o servicios previstos en el plan de empresa.

  Igualmente, no se deben descuidar los potenciales competidores en el futuro, es decir, qué empresas que todavía no están en el mercado podrían entrar en él; ni tampoco los de otras regiones geográficas. En el contexto actual y especialmente en los sectores relacionados con las tecnologías de la información y las comunicaciones, como el software libre, la competencia tiende a ser global y muchas empresas pueden ofrecer sus servicios directa o indirectamente en cualquier lugar.

- **Análisis de las barreras de entrada**. Las barreras de entrada son los obstáculos que toda empresa se encuentra al entrar en un nuevo mercado. Por ejemplo, la necesidad de una fuerte inversión en el caso de empresas de nueva creación o la ausencia de una marca establecida. Por ejemplo, el software libre se encuentra a menudo con la falta de calidad percibida, frente a empresas y soluciones de software propietario establecidas en el mercado.

  Claro que, de la misma manera, se pueden estudiar qué barreras de entrada se pueden favorecer una vez instalados en el mercado, para mantener alejada la competencia.

- **Influencia de las administraciones públicas**. El estudio de mercado debe tener en cuenta el modo en que las administraciones públicas, locales, regionales, nacionales o internacionales pueden afectar al mercado y, en consecuencia, a la viabilidad del plan de empresa. Así, las administraciones pueden actuar como reguladoras del mercado, pero también como proveedores y como clientes.

  Esto es particularmente cierto en el caso del software libre, que como se ha visto a lo largo de la asignatura es motivo de interés de múltiples administraciones, como la Junta de Extremadura o el Gobierno de Brasil.

  La realización de un estudio de mercado se debe planificar cuidadosamente y atraviesa diferentes fases, que se resumen en las siguientes:

  1) Recogida de información general, donde se obtiene un gran volumen de datos sobre el mercado objeto de estudio.

  2) Análisis de la información obtenida.

  3) Búsqueda selectiva de información, donde se obtiene la información ausente y necesaria para completar el estudio de mercado, que se habrá identificado tras el análisis previo.

  Para llevar a cabo la elaboración de un estudio de mercado se necesita una gran cantidad de información, que no siempre es de fácil acceso. Existen numerosos organismos y fuentes de información, tanto generales como especializadas en ciertas regiones o sectores: administraciones e institutos de estadística nacionales, administraciones regionales y locales, organismos privados como las cámaras de comercio o las asociaciones de empresas y revistas y publicaciones especializadas.

  Un buen estudio de mercado debe concluir con un análisis estratégico[19] que relacione los resultados del propio estudio con la descripción del negocio y los recursos previstos y que muestre el potencial del plan de empresa.

(19)A su vez, este análisis puede apoyarse en la utilización de herramientas estratégicas como el análisis DAFO (http://es.wikipedia.org/wiki/An%C3%A1lisis_DAFO) o de las 5 fuerzas de Porter (http://es.wikipedia.org/wiki/An%C3%A1lisis_Porter_de_las_cinco_fuerzas).

2.7. Plan de marketing

El objetivo del plan de marketing es la definición de las estategias comerciales que permitan alcanzar el volumen de facturación previsto en el análisis económico-financiero, que se verá en detalle en el apartado siguiente.

Por tanto, el plan de marketing detalla las acciones a llevar a cabo para explotar el modelo y la oportunidad de negocio descritos en el plan de empresa y aprovechar sus ventajas competitivas.

De este modo, el plan de marketing debe recoger los siguientes aspectos:

- **Estrategia comercial global**. La estrategia global debe definir de qué manera la parte comercial se integra en el proyecto empresarial. Se debe explicar cómo se van a identificar los clientes y cómo se contactará con ellos, cuáles son las motivaciones de los clientes al interesarse o decidirse por los productos o servicios ofrecidos y, en consecuencia, las características de los productos o servicios que se van a destacar para generar ventas, por ejemplo el precio, la calidad, la garantía y el soporte técnico, etc.
  El caso del software libre es bastante ilustrativo, ya que el principal reclamo para los potenciales clientes es la reducción de costes, y no tanto la calidad, que a menudo es superior a la del software propietario. En cambio, el cliente suele identificar los precios elevados del software propietario con una calidad superior, y el software libre, que resulta más económico, con una calidad inferior. Por ello, a la hora de realizar un plan de empresa basado en un modelo de negocio de software libre es esencial destacar la calidad superior[20] del software libre.

(20)Por ejemplo, poniendo en primer plano la interoperabilidad y la flexibilidad y la constante revisión y mejoras a las que está sometido el software libre.

- **Estrategia de ventas**: define cuáles son los objetivos de ventas a corto y largo plazo así como en qué sectores del mercado se van a introducir los productos o servicios ofrecidos en una primera fase y en el futuro. En cualquier caso, se deben justificar adecuadamente las decisiones, apoyándose en los resultados del estudio de mercado.

- **Estrategia de precios**: determina en primer lugar los precios con los que se van a comercializar los productos y servicios ofrecidos, comparándolos, si es posible, con los de los competidores, con una estimación del margen bruto de beneficios y una evaluación para determinar si este margen es suficiente para soportar toda la actividad empresarial[21].

(21)Igualmente, es recomendable comparar los márgenes propios con los de la competencia, en el caso de que se disponga de tal información.

Es muy importante justificar la política de precios, sobre todo en comparación con la de los competidores. En el caso de que el precio de los productos o servicios ofertados sea superior a los de la competencia, se debe explicar en términos de novedad y calidad, prestaciones y garantías superiores. En el caso de que el precio sea inferior, se debe justificar cómo se mantiene la rentabilidad, por ejemplo gracias a una mayor eficiencia y menores costes de producción. De nuevo, es muy importante explicar las causas del bajo coste del software libre y los beneficios que lleva aparejados.
Finalmente, hay que tener en cuenta que la estrategia de precios debe ser la óptima, es decir, aquella que maximice el margen de beneficios y por tanto la rentabilidad. En ocasiones, un precio más elevado, a pesar de reducir en parte las ventas, puede acarrear mayores beneficios.

- **Política de ventas**: recoge la composición, forma de contratación y perfil del equipo comercial o de ventas, incluyendo comerciales y representantes, en el momento de puesta en marcha de la empresa y su evolución a medio y largo plazo. Esto incluye la política de márgenes comerciales, las medidas de promoción que se van a ofrecer a representantes, comerciales y también distribuidores autorizados.
  Forman también parte de la política de ventas: la estimación de las ventas de cada comecial y representante, los incentivos, los períodos de cobro acordados a los clientes, las promociones especiales como descuentos, anticipos, rappels, etc.

- **Promoción y publicidad**: debe describir las medidas que se van a poner en práctica para atraer la atención de clientes potenciales hacia los productos o servicios ofrecidos. Estas medidas incluyen pasar por buzoneos electrónicos, participación en ferias y eventos comerciales, publicidad en sitios web, etc.
  Finalmente se debe cuantificar el coste de la promoción y su retorno, en consultas de clientes y ventas cerradas.

- **Servicio posventa y garantías**: debe describir el servicio posventa y las garantías que se ofrecen a los productos o servicios ofrecidos, cuando sea aplicable. Es decir, qué tipo de servicio y garantía se ofrece, su duración temporal, su precio en el caso de que sea optativa y sus costes para la empresa.
  En el caso del software libre, una parte del servicio posventa está proporcionado indirectamente por la comunidad de desarrolladores y usuarios, que en proyectos de éxito mejoran constantemente el producto. El plan de empresa basado en software libre debe tener esto en cuenta y presentarlo como una ventaja, pero nunca como el único soporte añadido. Hay que tener en cuenta que la inmensa mayoría de los clientes desean que el servicio posventa esté incluido y garantizado en las condiciones de la venta.

Por último, se debe valorar también la importancia del servicio posventa y de las garantías ofrecidas en la decisión final del cliente, y comparar el servicio propio con el proporcionado por los competidores.

- **Política de distribución**: La política de distribución debe describir los canales de distribución que se van a utilizar y las políticas de descuentos, comisiones y márgenes asignados a cada uno de estos canales.
  En los modelos de negocio de software libre encontramos con frecuencia la existencia de programas para empresas asociadas[22] bajo distintas formas: integradores de sistemas, vendedores de software, etc., y a las que se les ofrece comisiones y se les facilitan servicios y asistencia dedicada y acceso a canales de promoción.
  Por otra parte, como se ha comentado anteriormente, los productos y servicios de software libre pueden ofrecerse a menudo sin problemas en el mercado global, por lo que el plan de marketing debe estudiar esta posibilidad y las particularidades que presentaría, incluyendo el efecto de las leyes internacionales en la actividad de la empresa, la gestión de los cobros en el extranjero, etc.

[22] Por ejemplo, el programa de asociados del ERP basado en software libre Openbravo, que podéis consultar en http://www.openbravo.com/partners/join-openbravo/details/.

2.8. Análisis económico-finaciero

El análisis o estudio económico-financiero es también uno de los elementos básicos de todo plan de empresa, ya que su objetivo es evaluar la viabilidad y el potencial económico del proyecto empresarial, detectar las necesidades de inversión para su puesta en marcha, identificar los recursos disponibles inicialmente y presentar las distintas posibilidades de financiación.

Contrariamente a lo que podría parecer, el análisis económico-finaciero es una de las partes más creativas de la elaboración de un plan de empresa.

Los estados finacieros o aspectos fundamentales que debe cubrir el análisis económico-finaciero son los siguientes:

- Estado de la tesorería durante el primer año desglosada por meses, a fin de reflejar los efectos de la estacionalidad[23].

- Análisis del fondo de maniobra, que permite conocer la liquidez patrimonial de la empresa.

[23] Incluso planes de empresa con un fuerte componente tecnológico, como los basados en software libre, se ven afectados por la estacionalidad de la economía, por ejemplo durante las vacaciones de verano.

- Cálculo del punto de equilibrio y alternativas en el caso de que el volumen de ventas objetivo no se alcanzara.

- Necesidades y alternativas de financiación, elegiendo las más rentables y aportando los elementos que justifiquen la decisión.

- Balances anuales a cinco años vista, con el primer año desglosado por meses.

- Origen y aplicación de los fondos, que permite pronosticar situaciones de riesgo para la empresa y evaluar la procedencia y utilización de fondos a largo plazo.

Fondo de maniobra

El fondo de maniobra mide el equilibrio patrimonial de una entidad, ya que acredita la existencia de activos líquidos en mayor cuantía que las deudas con vencimiento a corto plazo.

Es conveniente realizar un análisis conjunto de estos estados financieros y obtener unas conclusiones que aporten información sobre el proyecto empresarial en su conjunto: la cantidad de capital necesario y cuándo será necesario, y la deuda necesaria y cuándo se debe pagar esta deuda, entre otros.

Igualmente se debe explicar la rentabilidad esperada de la inversión y cuándo se recuperará esa inversión.

Como se comentó anteriormente, hay que evitar caer en la tentación de presentar un análisis económico-finaciero demasiado optimista para ganarse la confianza de los inversores, ya que tarde o temprano se volverá contra la propia empresa y pondrá en entredicho su viabilidad y credibilidad.

2.9. Forma legal

Si el objetivo último del plan de empresa es efectivamente la creación de una nueva empresa[24], se debe elegir la forma legal con la que ésta se va a constituir, su régimen fiscal y sus socios fundadores. Del mismo modo, se debe recoger el nombre de todos los socios e inversores y su participación en la nueva sociedad.

(24)En el caso de que se tratara de un plan de negocio para una empresa ya constituida, este apartado describiría la naturaleza jurídica de ésta y las posibles modificaciones que la implementación del plan de negocio podría llevar consigo.

Es conveniente detallar paso a paso todos los trámites necesarios para constituir la nueva empresa, así como su coste y el tiempo necesario para su ejecución. Se debe también especificar si se va a recurrir a servicios de asesoría externa especializada y su coste.

2.10. Gestión de riesgos

Todo proyecto empresarial, ya se trate de la creación de una nueva empresa o de una nueva línea de negocio, implica numerosos riesgos que en ocasiones son ineludibles. Por ello, el plan de empresa debe incluir una descripción completa de los riesgos y de sus consecuencias.

Ved también

En el apartado "Gestión de riesgos" de la primera unidad podréis encontrar una introducción general a este tema.

Los riesgos pueden clasificarse según sean internos, con origen en la propia empresa, o externos, y según al área funcional que afecten: técnica, comercial, etc.

Por ejemplo, riesgos internos pueden ser la existencia de retrasos en la producción o falta de personal cualificado, mientras que riesgos externos puede ser una nueva regulación del mercado, que reduzca en parte la rentabilidad o la aparición de nuevas tecnologias que dejen obsoletos los productos o servicios ofrecidos.

Para cada riesgo se debe definir un plan de contingencia, lo que incluye una serie de acciones de prevención[25], es decir, para evitar que el riesgo llegue a realizarse, y una serie de acciones de mitigación o remedio[26], qa adoptar en el caso de que el riesgo se realice.

[25]Por ejemplo, para prevenir la aparición de nuevas tecnologías que pudieran dejar obsoletos los productos o servicios previstos en el plan de empresa se debería poner en práctica una vigilancia tecnológica activa y, eventualmente, colaborar con empresas u organizaciones que trabajen en la misma área.

Hay que tener en cuenta que algunos riesgos pueden provocar efectos negativos, pero también positivos. Por ejemplo, cambios en el marco legal o político que pueden afectar al modelo de negocio, pero al mismo tiempo dar lugar a nuevas oportunidades de negocio.

[26]Por ejemplo, se podrían asignar recursos humanos y materiales provenientes de otros departamentos a fin de recuperar el retraso en la producción.

La correcta identificación y evaluación de los riesgos en un proyecto empresarial y la preparación de planes de contingencia adecuados, antes que denotar debilidades del proyecto, destacan las habilidades de gestión y de previsión de los promotores y aumentan su credibilidad.

2.11. Resumen y evaluación

El último apartado del plan de empresa debe resumir los puntos fuertes y débiles del proyecto empresarial, las ventajas y oportunidades que ofrece y las principales amenazas y riesgos.

El resumen es la última oportunidad para convencer a un inversor potencial, así que se debe ser muy convincente y aprovecharlo para reforzar los argumentos a favor del proyecto empresarial y en los que creen sus promotores.

Sin embargo, puede darse la posibilidad de que tras la elaboración del plan de empresa los propios promotores del proyecto descubran que éste no sea tan rentable como se esperaba, o incluso completamente inviable. Esto muestra la utilidad del plan de empresa como herramienta para la identificación de las mejores oportunidades de negocio.

2.12. Plan de empresa y software libre

La elaboración de un plan de empresa basado en software libre no presenta grandes diferencias respecto a los planes de empresa de otros sectores y algunas de sus particularidades ya se han presentado a lo largo de los apartado anteriores.

En general, hay que recordar que un plan de empresa puede estar dirigido a diferentes tipos de lectores: asesores, inversores, técnicos, banqueros. Por ello se debe emplear un lenguaje comprensible por todos ellos y evitar la utilización de un vocabulario demasiado técnico. Cuando la utilización de estos términos sea inevitable, es recomendable explicar bien cada uno de los conceptos en un lenguaje accesible. Un inversor nunca invertirá en algo que no comprende del todo.

Igualmente, hay que prestar atención a explicar las particularidades del software libre, haciendo hincapié en las diferencias respecto al software propietario y sus principales ventajas. No hay que dudar en recurrir a ejemplos reales y casos de éxito que refuercen los argumentos presentados en el plan de empresa.

Por otra parte, si bien el software libre comienza a tener un papel cada vez más relevante en los medios y en la sociedad gracias al compromiso de la comunidad de software libre, de empresas y de administraciones públicas, su naturaleza y sus implicaciones económicas no son tan conocidas.

De nuevo, hay que prestar mucho cuidado a explicar correctamente los modelos de negocio basados en software libre y estar preparado para responder e incluso anticipar las preguntas más frecuentes. Por ejemplo, ¿cómo se puede invertir y ganar dinero en algo que cualquiera puede copiar?

3. Producción de software libre

Buena parte de los modelos de negocio presentados en el primer apartado dependen, en mayor o menor medida, del desarrollo de software libre.

Uno de los problemas a la hora de hablar de proyectos de software libre es que tan solo los proyectos exitosos tienen repercusión en la comunidad y sólo los muy exitosos llegan a los medios no especializados.

Sin embargo, antes de abordar la producción de software libre hay que recordar que la inmensa mayoría de proyectos de software libre son un fracaso, por unas razones o por otras. Puede deberse simplemente a que el proyecto no consigue producir un software de calidad y competitivo, o bien porque no consigue atraer la atención de la comunidad de desarrolladores y usuarios.

Por supuesto, y como se ha visto en los materiales de la asignatura, no hay que olvidar que un proyecto de software libre debe tratarse como un proyecto de software, y en último lugar, simplemente como un proyecto de ingeniería. Por ello, todo proyecto de software libre presenta en primer lugar los mismos riesgos y problemas que cualquier otro proyecto.

Sin embargo, y dada la naturaleza libre de este tipo de proyectos, hay también otras fortalezas y debilidades que conviene conocer. Por el carácter aparentemente no profesional de muchos proyectos de desarrollo de software libre puede parecer que la ejecución de éstos, respecto a la de los proyectos de desarrollo de software tradicionales, sea más sencilla. Nada más lejos de la realidad.

El objetivo de este apartado es presentar las particularidades de los proyectos de desarrollo de software libre respecto al desarrollo de software propietario y mostrar una serie de buenas prácticas que faciliten su éxito. Estas prácticas se corresponden con las principales áreas y elementos necesarios para poner en marcha y ejecutar un proyecto de software libre, que son las siguientes:

- Creación y presentación del proyecto
- Infraestructura necesaria
- Organización de la comunidad
- Desarrollo
- *Releasing* y *packaging*

- Elección de licencias

Por supuesto, no todos estos pasos son obligatorios. Como se ha visto en los modelos de negocio, una empresa dedicada al software libre puede ser la iniciadora del proyecto, o bien, en la mayoría de los casos, unirse a un proyecto ya existente.

Esta última opción es con frecuencia la más recomendable y, de hecho, dada la naturaleza del software libre, no excluye la posibilidad de que a partir de un proyecto existente se cree uno nuevo con la identidad de la empresa o la organización interesada en dirigir su desarrollo.

3.1. Creación y presentación del proyecto

Este apartado se ocupa fundamentalmente de los pasos necesarios para crear un nuevo proyecto de software libre y presentarlo a la comunidad.

Así, el primer paso antes de crear un nuevo proyecto es descubrir si existe algún proyecto que realice al menos en parte lo que se pretende. Si hay un proyecto similar de software libre al que se puede contribuir, o que se puede reaprovechar para poner en marcha el nuevo proyecto, es conveniente ponerse en contacto con sus responsables para explorar las posibilidades de colaboración y sus planes futuros.

Buscadores genéricos

Los buscadores genéricos son la primera etapa para descubrir proyectos existentes, así como los sitios de noticias, los directorios y las forjas públicas como http://freshmeat.net, http://directory.fsf.org y http://www.sourceforge.net.

Si se ha decidido crear un nuevo proyecto, lo primero que hay que hacer es elegir un nombre que lo identifique en la comunidad. Como regla general, un buen nombre debe dar una idea de qué hace el software, o al menos de su campo de aplicación, y debe ser fácil de recordar.

Para bien o para mal, el inglés es la lengua oficial de facto en Internet. Por ello, si el proyecto busca tener un impacto global, y así debería ser en la mayoría de los casos, es conveniente que el nombre tenga cierto significado en inglés, o bien que sea neutro[27].

(27)Es decir, que sea un nombre común a varias lenguas, como Apache, o que no se asocie a otra lengua mayoritaria, como por ejemplo Ubuntu.

Por otra parte, se debe prestar atención a los aspectos legales, de manera que el nombre no entre en conflicto con marcas registradas y que los potenciales dominios de alto nivel[28] en Internet asociados estén disponibles.

(28)Es decir, los dominios .com, .net y .org.

De la misma manera que se ha visto en el apartado dedicado a la creación de un plan de empresa, todo proyecto debería contar con una definición clara de su misión, que atraiga la atención de usuarios y desarrolladores y les permita decidir si están interesados en el proyecto, o no.

Junto a la misión, es igualmente importante identificar inequívocamente el proyecto como software libre, lo cual implica hacer una referencia clara a software libre (*free software*) o software de código abierto (*open-source software*).

Otros elementos importantes a la hora de presentar un proyecto de software libre son:

- **Lista de funcionalidades previstas[29] y requisitos actuales**. Redactada de una manera sencilla, es decir, sin tecnicismos. Es, en cierto modo, un resumen detallado de lo que hace el software, que permita a los usuarios saber fácilmente si se trata de las funcionalidades que buscan.
  De igual modo, los requisitos deben ser fáciles de entender, de manera que el usuario sepa si puede instalar y utilizar la aplicación en su sistema.

(29)Se pueden indicar con una mención "en progreso" o "en desarrollo", idealmente con la fecha o la versión en que estarán disponibles.

- **Estado de desarrollo**. En la comunidad de software libre los usuarios suelen estar muy interesados en saber cómo avanza el desarrollo del proyecto, tanto si se trata de un nuevo proyecto como de uno ya maduro. Para ello deben explicarse los objetivos del proyecto a corto y largo plazo, las funcionalidades en las que se está trabajando actualmente y que estarán disponibles en futuras *releases*, etc.

- **Descargas disponibles**. El código fuente debe poder descargarse siempre en formatos estándar, de una manera sencilla, que no suponga ninguna complicación para el usuario[30].
  El proceso de instalación también debe ser sencillo y, sobre todo, de acuerdo a los estándares desde el principio del proyecto. Igualmente, en un primer momento no es necesario facilitar paquetes binarios o ejecutables, a menos que el proceso de compilación sea muy complejo.

(30)Es conveniente, por ejemplo, evitar procesos de registro de usuarios para acceder a la sección de descargas.

- **Repositorio de desarrollo**. Los potenciales desarrolladores, al contrario que los usuarios, están más interesados en acceder al repositorio de trabajo, donde se puede seguir la evolución del proyecto día a día y participar en ella, ya sea añadiendo nuevas funcionalidades o corrigiendo errores. Para ello es conveniente que todos puedan acceder a la lectura del repositorio, mediante un acceso anónimo.

- **Seguimiento de errores**. Al igual que el repositorio, la base de datos de seguimiento de errores[31] debe estar también abierta a todos. Paradójicamente, un proyecto es mejor cuantos más errores contiene su base datos, ya que esto implica un mayor número usuarios y una mayor participación de los usuarios en el proyecto.
  Al inicio del proyecto, el número de errores será muy bajo. Una buena práctica es registrar en la base de datos los errores resueltos internamente por el equipo que pone en marcha el proyecto.

(31)Con frecuencia se hace referencia a los términos *bug tracker* o *bug database* en inglés.

- **Canales de comunicación**. Uno de los objetivos de todo proyecto de software libre es crear una comunidad alrededor de él y, para que dicha comunidad se organice, es necesario facilitar los canales de comunicación adecuados. Esto incluye listas de correo, canales de IRC, foros, etc.
  En una primera fase del proyecto es conveniente no diversificar ni especializar en exceso los canales de comunicación. Un único foro o lista de distribución para usuarios y desarrolladores puede ser suficiente y favorecer las interacciones entre ellos.

- **Documentación para usuarios y desarrolladores**. La documentación es esencial en todo proyecto de software libre, tanto para los usuarios como para los desarrolladores.
  Una buena documentación para el usuario debe explicar a éste cómo instalar el software y cómo utilizar sus funciones. Se le puede facilitar también un pequeño tutorial, que le enseñe a realizar las tareas más frecuentes paso a paso. Un excelente complemento de la documentación es el mantenimiento de una sección de preguntas frecuentes o FAQ.
  La documentación para desarrolladores debe incluir la información de contacto de los principales desarrolladores del proyecto, las instrucciones para enviar informes de errores y parches, así como una presentación sobre la organización del desarrollo y la toma de decisiones entre los desarrolladores.

Todos estos elementos se verán en detalle en los apartados siguientes.

Para concluir este apartado cabe destacar que la apariencia –es decir, cómo la comunidad de software libre percibe el proyecto– tiene un papel bastante importante en el éxito o el fracaso de un proyecto de software libre.

Muchos desarrolladores no prestan suficiente atención a esta tarea de comunicación y relaciones públicas, la cual, sin embargo, es un elemento indispensable en prácticamente todos los proyectos de software libre de éxito.

Para ello hay que definir claramente cuales son los objetivos del nuevo software, que normalmente se pueden resumir en:

- **Definir claramente qué hace el software**: sus funcionalidades principales, el estado actual de desarrollo y los planes de futuro, además de su posicionamiento respecto a las soluciones y los proyectos ya existentes.

- **Dar a conocer el software**: hacerlo llegar a la comunidad o al mercado de usuarios y desarrolladores potencialmente interesados.

- **Potenciar el uso del software**: que los usuarios y los desarrolladores potenciales sepan utilizar el nuevo software y lo adopten frente a las soluciones alternativas.

- **Involucrar nuevos desarrolladores en el proyecto**: que éstos contribuyan al desarrollo del proyecto mediante la implementación de nuevas funcionalidades y que aporten sus puntos de vista sobre la dirección que debería tomar el proyecto en el futuro.

Los dos últimos objetivos, conseguir muchos usuarios y muchos desarrolladores, son a menudo los más importantes. Sin embargo, es necesario poner en práctica una estrategia para los usuarios y otra diferente para los desarrolladores, los cuales, aun perteneciendo a la misma comunidad de software libre, representan audiencias muy distintas.

Se debe definir claramente el mensaje que quiere hacerse llegar a cada uno de ellos y estructurarlo con una complejidad progresiva, de modo que el nivel de detalle ofrecido se corresponda con el esfuerzo requerido por parte del lector. Por ejemplo, no tiene sentido saturar al usuario con la arquitectura del software, ni tampoco adentrar a los desarrolladores en detalles técnicos sin antes dar una visión adecuada de la arquitectura.

Finalmente, este mensaje debe ser fácilmente accesible y puede hacerse llegar a sus destinatarios mediante anuncios en foros o comunidades relacionadas, en el sitio web del proyecto e incluso en la documentación, entre otros.

3.2. Infraestructura

Cualquier proyecto de software libre necesita una serie de herramientas que permitan gestionar la información que se genera en el día a día del proyecto, desde el código desarrollado hasta las comunicaciones entre sus miembros.

Algunas de estas herramientas ya se han introducido en el apartado anterior, pues son esenciales para poner en marcha el proyecto:

- **Sitio web**
  Proporciona una fuente centralizada de información sobre el proyecto y da acceso a otras herramientas de gestión especializadas.

- **Listas de correo**
  Es uno de los canales de comunicación utilizados con más frecuencia en los proyectos de software libre. Los intercambios de mensajes suelen quedar archivados y se utilizan como referencia y base de conocimiento del proyecto.

- **Sistema de control de versiones**
  Permite que los desarrolladores controlen la creación y la gestión de código, volviendo a versiones anteriores y uniendo diferentes versiones. Gra-

cias al sistema de control de versiones, cualquiera puede visualizar el estado actual del código, así como su evolución en el tiempo.

- **Sistema de seguimiento de errores**
  Permite que los desarrolladores realicen un seguimiento de las funcionalidades y los errores en los que está trabajando cada uno, se coordinen entre ellos y planifiquen las próximas releases. Si bien el seguimiento de errores es su función principal, la base de datos puede utilizarse igualmente para realizar el seguimiento de cualquier tarea del proyecto, como las nuevas funcionalidades.
  Gracias al sistema de seguimiento de errores, cualquiera puede saber si un determinado error ha sido solucionado o si alguien está trabajando en él. Junto al sistema de control de versiones, permite conocer el dinamismo y la actividad registrada del proyecto.

- ***Chat* o canal de conversación**
  Proporciona un canal de comunicación donde resolver dudas y problemas rápidamente. Las conversaciones no se suelen archivar, por lo que es aconsejable que las discusiones más complejas tengan lugar en listas de correo.

Cada una de estas herramientas responde a unas necesidades concretas, principalmente de comunicación y de gestión de la información. La experiencia y las características de la comunidad de usuarios y desarrolladores alrededor del proyecto van a dictar la configuración y el uso de estas herramientas. No obstante, vale la pena comentar algunos aspectos que pueden resultar de utilidad para la mayoría de proyectos de software libre.

Las listas de correo son un elemento esencial de todo proyecto de software libre, por lo que hay prestar especial atención a su gestión y uso. Es prácticamente obligatorio disponer de un sistema gestor de listas de distribución, cuya configuración y mantenimiento pueden resultar complicados en un principio.

Recursos web

Entre los sistemas más populares están Mailman (http://www.list.org), Smartlist (http://www.procmail.org), Ecartis, Listproc (http://listproc.sourceforge.net) o Ezmlm.

Las funcionalidades y las opciones principales de un sistema gestor de listas de distribución son las siguientes:

- Suscripción mediante correo electrónico y mediante una interfaz web
- Suscripción en modo *digest* o normal[32]
- Moderación
- Interfaz de administración
- Configuración de las cabeceras de los mensajes
- Gestión y consulta de archivos

(32)En el modo *digest* se recibe una compilación de todos los mensajes periódicamente, normalmente cada mes o cada semana, mientras que en el modo normal los mensajes se reciben inmediatamente.

Por otra parte, las listas de correo se pueden integrar con otras herramientas tales como el sistema de control de versiones o el sistema de seguimiento de errores para notificar, por ejemplo, cambios en el código fuente o modificaciones en el estado de errores y tareas en curso.

Igualmente, el sistema de control de versiones es un elemento indispensable para cualquier proyecto de software libre que aspire a articular una comunidad de desarrolladores. El funcionamiento de casi todos los sistemas de control de versiones se basa en la existencia de una copia remota, compartida por todos los desarrolladores y de la que se pueden consultar todas las versiones. Cada desarrollador dispone de una copia local de esa copia remota, sobre la que trabaja. Puntualmente, cada desarrollador envía sus modificaciones a la copia remota, compartiéndolas con el resto.

Las funcionalidades principales de un sistema de control de versiones son:

- ***Commit***: integrar los cambios de la copia local en la copia remota, los cuales quedan así registrados en la base de datos de control de versiones.

- ***Update***: integrar los cambios de los demás desarrolladores en la copia local.

- ***Checkout***: obtener una copia local a partir de la copia remota.

Vale la pena destacar que, dentro del proyecto, todo documento o archivo editado puede y debe ser objeto de un control de versiones, por lo que éste no se debería limitar a los archivos de código fuente. La utilización de un sistema de control de versiones puede resultar muy práctica para editar y compartir documentación o informes técnicos y, en general, para cualquier documento que haya sido creado y mantenido en colaboración.

Como se ha comentado anteriormente, el sistema de seguimiento de errores permite realizar muchas otras funciones, además de la que indica su nombre. Esto incluye el seguimiento de cualquier tipo de tarea, como la implementación de funcionalidades nuevas, la preparación de releases o el soporte a los usuarios.

El ciclo de vida de un error suele ser el siguiente:

- **Notificación del error**: Todo error incluye al menos un resumen y una descripción inicial que contienen, a ser posible, los elementos necesarios para reproducir el error. La mayoría de sistemas de seguimiento de errores permite configurar campos específicos. No hay que olvidar que los errores pueden venir tanto de la comunidad de usuarios como de la de desarrolladores.
  Una vez archivado, el error queda en estado abierto y no está asignado a nadie. Durante ese tiempo, la gente que acceda a la base de datos puede

leer la descripción del error y, eventualmente, pedir más información al usuario o al desarrollador que lo ha notificado.

- **Reproducción del error**: Siguiendo las indicaciones de la descripción del error, alguien consigue reproducirlo, por lo que éste queda validado. Es decir, se puede decir que el error es auténtico.

- **Diagnóstico del error**: Durante las fases anteriores puede ocurrir que un desarrollador se haga responsable de la resolución del error, o bien que alguien con autoridad dentro del proyecto lo asigne al desarrollador más adecuado.

- **Asignación del error**: durant les fases anteriors pot passar que un desenvolupador es faci responsable de la resolució de l'error, o bé que algú amb autoritat dins del projecte l'assigni al desenvolupador més adequat.
  Es esencial notificar esto en la base de datos, a fin de evitar que dos desarrolladores estén trabajando en la resolución del mismo error sin saberlo. Es posible notificar igualmente la fecha esperada de resolución, o la *release* en la que el error se habrá solucionado.

- **Resolución del error**: Una vez el desarrollador resuelve el error, lo marca como cerrado o resuelto.

A veces los errores se resuelven rápidamente, por lo que algunas de estas fases se pueden obviar. O incluso puede darse que el error no sea tal y que se deba a un mal uso por parte del usuario. En cualquier caso, por sencilla que sea la solución, siempre es conveniente registrar el error y comunicárselo adecuadamente al usuario.

Otra situación frecuente se da cuando varios usuarios notifican el mismo error, lo que se conoce como errores duplicados. En este caso es conveniente agrupar todas las notificaciones bajo una única notificación, lo que permiten concentrar esfuerzos y disponer de toda la información en el mismo lugar.

Finalmente, puede ocurrir que un error considerado como resuelto no lo esté en realidad, generalmente porque el patrón de reproducción seguido no coincide con el proporcionado por el usuario que ha notificado el error. En este caso, el usuario puede reabrir el error, aportando siempre toda la información necesaria. Existen numerosas forjas públicas que ofrecen estas y otras herramientas, listas para ser utilizadas en los proyectos de software libre. Estas plataformas ofrecen una serie de ventajas y desventajas.

Web complementarias

Entre las forjas públicas más populares están SourceForge (http://www.sourceforge.net), Savannah (http://savannag.gnu.org o BerliOS.de (http://www.berlios.de). Además, algunas organizaciones ofrecen alojamiento a proyectos dentro de su área de interés, como Apache (http://www.apache.org) o Tigris (http://www.tigris.org).

Entre sus ventajas cabe destacar su capacidad y el ancho de banda disponible: no importa el posible éxito del proyecto, los servidores siempre van a estar en funcionamiento. Vale la pena recordar el trabajo adicional que supone mantener un servidor de alta disponibilidad en marcha. Por otra parte, las herramientas proporcionadas por estas forjas están ya configuradas y normalmente son muy sencillas de utilizar. Evidentemente, la desventaja principal es que la flexibilidad y las posibilidades de configuración de las herramientas ofrecidas son limitadas.

Así, al comenzar el proyecto puede ser recomendable albergarlo en una forja pública, pero a la vez dejar abierta la posibilidad de disponer de un alojamiento propio en el futuro, empezando por registrar el nombre del dominio asociado al proyecto. Por ejemplo, el hecho de disponer de un sitio web informativo del proyecto, que redirija a una forja pública para los aspectos ligados al desarrollo del código puede ser, aunque no una solución óptima, sí un buen compromiso.

3.3. Organización de la comunidad

Una de las mayores diferencias entre los proyectos de software libre y los proyectos de software propietario es el modo de organizarse de la comunidad de desarrolladores.

En un proyecto de software propietario, la organización corresponde normalmente a la organización jerárquica del equipo o el departamento que, dentro de una empresa, lleva a cabo a cabo el desarrollo. Aunque en un proyecto de software libre también se aprecia a veces cierta jerarquía, parcialmente basada en los méritos de cada desarrollador, la organización de la comunidad de desarrolladores es más flexible pero también más fuerte.

Paradójicamente, uno de los motivos que hacen que la comunidad de desarrolladores trabaje y se mantenga unida es la posibilidad de crear un nuevo proyecto independiente[33] a partir del proyecto original. La posibilidad de que un proyecto de software libre se escinda es normalmente perjudicial, tanto para los desarrolladores como para los usuarios. Es precisamente esta amenaza lo que hace que la comunidad se organice y se esfuerce para tomar decisiones conjuntamente.

(33) En inglés, *forkability*, es decir la posibilidad de realizar un *fork*.

Dicho de otro modo, el hecho de que exista la posibilidad de escisión hace que la comunidad busque un consenso más o menos democrático en las grandes decisiones del proyecto.

En general, existen dos formas de organización de las comunidades de software libre, si bien la mayoría de los proyectos acaban adoptando una posición intermedia entre ambas. Estas dos formas son las siguientes:

- **Organización basada en un "dictador benevolente".**
  El dictador benevolente es una figura con autoridad para tomar decisiones definitivas, de importancia para la vida del proyecto. Sin embargo, a menudo el dictador benevolente no toma las decisiones directamente, sino que suele actuar más bien de árbitro en las discusiones, tratando de compatibilizar los puntos de vista de los desarrolladores e identificar las aportaciones más valiosas. Otra forma de actuación del dictador benevolente es delegar en expertos que puedan ocuparse de las decisiones o discusiones en marcha. El dictador benevolente suele ser un desarrollador con suficiente experiencia en el proyecto y en las tecnologías relacionadas, pero que no es necesario que sea el más experto. Basta con que sea capaz de entender el proyecto en su totalidad y reconocer las contribuciones de mayor calidad.

- **Organización basada en el consenso.**
  Por consenso se entiende los acuerdos que toda la comunidad acepta más o menos tácitamente, es decir, cuando nadie se opone a las decisiones ni a la dirección que se van tomando en el proyecto, por lo que el proceso de consenso no suele ser en absoluto formal. No obstante, esto no impide que cuando no se alcanza un consenso en un determinado tema, se pueda realizar una votación.
  La mayoría de discusiones en la vida de un proyecto suele ser de naturaleza técnica, por lo que el consenso se produce cuando todo el mundo esta de acuerdo en, por ejemplo, el diseño o la implementación de una funcionalidad o en la manera de resolver un error. En estos casos, además, un miembro suele hacer un resumen de la discusión al final de ésta.

En general todas las comunidades, y especialmente las basadas en el consenso, tienen un excelente apoyo en el sistema de control de versiones, el cual permite volver atrás y deshacer cualquier decisión que se revele equivocada.

Los proyectos suelen comenzar con una organización basada en un dictador benevolente y, a medida que la comunidad crece, se mueven hacia una organización más basada en el consenso. Esto suele ocurrir en ciertos momentos de la vida del proyecto, como por ejemplo cuando el dictador benevolente abandona su posición y su autoridad se diluye en la comunidad, especialmente entre los miembros más respetados de ésta.

Al cabo de un tiempo, las convenciones y los acuerdos tomados por una comunidad mediante consenso pueden ser muy grandes, por lo que es conveniente recoger los elementos principales en un documento que sirva de guía y referencia en el futuro. Éste puede incluir tanto la forma de gobierno de la comunidad, como las convenciones y las recomendaciones para los desarrolladores.

Recursos web

Podéis consultar las guías del proyecto Subversion o de la Fundación Apache (http://www.apache.org/foundation/how-it-works.html y http://www.apache.org/foundation/voting.html).

Finalmente cabe preguntarse cuál es el papel de las empresas en las comunidades de software libre.

Por una parte podemos considerar el caso de una empresa que desee iniciar un proyecto de software libre y crear una comunidad de usuarios y desarrolladores. Por otra parte, puede darse el caso de una empresa que se una a un proyecto de software libre ya en marcha. En cualquiera de los dos casos, la empresa debe definir claramente sus objetivos respecto al proyecto de software libre y cuál va a ser su participación en la comunidad.

Las posibilidades son muy variadas. Por ejemplo, la empresa puede buscar una posición de liderazgo en la comunidad y dirigir el proyecto o, sencillamente, tener voz en las discusiones, participar activamente en la implementación de nuevas funcionalidades o dedicar solamente algunos de sus desarrolladores a resolver los problemas de sus clientes.

Sin olvidar la gran dificultad que supone construir un proyecto exitoso de software libre, resulta evidente que, al menos para los proyectos iniciados por una empresa, la comunidad ya existe: es la formada por los desarrolladores de la empresa y sus clientes.

En esta situación, el modelo de organización basado en el dictador benevolente parece el más adecuado, al menos al principio, pero es necesario definir las reglas de participación de la comunidad. El reto es, por una parte, conseguir que estos clientes se conviertan en usuarios activos y que participen en la mejora del proyecto y, por otra parte, conseguir que otros desarrolladores se involucren.

La solución, si bien difícil, pasa por ofrecer ventajas o algún tipo de valor añadido a los usuarios y a los desarrolladores que participen en la comunidad.

Una buena práctica consiste en que el equipo de desarrollo de la empresa trabaje complemente integrado en la comunidad y siguiendo la metodología de desarrollo del proyecto de software libre. Esto implica que la participación de los desarrolladores en el proyecto debe ser duradera, a fin de familiarizarse con el funcionamiento de la comunidad y ganar credibilidad dentro de ella.

3.4. Desarrollo

En este apartado se presenta el proceso de desarrollo de un proyecto de software libre, no desde el punto de vista técnico, que dependerá de la naturaleza de cada proyecto, sino desde el punto de vista de la gestión del proyecto y la coordinación de los desarrolladores.

En lo que respecta al desarrollo, hay que tener en cuenta que una de las diferencias de los proyectos de software libre respecto a los proyectos de software propietario es la ausencia de una organización centralizada. Por ejemplo, cuando se aproxima la fecha de una nueva release, una empresa puede dedicar un número arbitrario de recursos a su preparación. En cambio, los desarrolladores voluntarios que forman la comunidad no son tan fáciles de dirigir. Las motivaciones de cada uno de ellos son diferentes y, si bien algunos pueden estar interesados en publicar una nueva *release* a tiempo, otros pueden estar interesados solamente en alguna funcionalidad concreta.

De este modo, la distribución de tareas en un proyecto de software libre se basa fundamentalmente en la independencia entre éstas y la regla general es que cada desarrollador trabaja en lo que desea y cuando lo desea.

No obstante, esta aproximación no deja de ser en parte ideal y, en la mayoría de proyectos de software libre, resulta necesario el trabajo de una persona o equipo que coordine a todos los desarrolladores voluntarios. Por ejemplo, este equipo puede estar formado explícitamente por los iniciadores del proyecto o el dictador benevolente o, implícitamente, por los miembros con más experiencia y ascendencia sobre la comunidad.

Algunas de las tareas principales de coordinación, necesarias para la buena marcha del proyecto, son:

- **Delegar**: una de las tareas principales de los coordinadores del proyecto es delegar tareas en otros desarrolladores. Cuando alguien delega una tarea en otra persona, y ésta acepta, el beneficio es doble: el coordinador encuentra alguien que hace el trabajo por él y la otra persona, en cambio, ve reconocido su trabajo, en tanto que éste le ha sido confiado. Por ello, la mejor manera de delegar una tarea es a través de un canal de comunicación visible para toda la comunidad y dando siempre la opción de declinar la oferta.
  En este caso, el coordinador debe conocer las habilidades y los intereses de los miembros de la comunidad y, en función de ello, dirigir sus demandas. Por ejemplo, no tiene sentido pedir algo a alguien que no tiene la capa-

cidad necesaria para llevarlo a cabo, ni tampoco a alguien que ya realiza numerosas tareas.

- **Realizar críticas y elogios**: la valoración adecuada de las contribuciones de cada uno de los desarrolladores del proyecto tiene un papel muy importante en la creación de un ambiente cordial dentro de la comunidad y, por supuesto, las valoraciones emitidas por los coordinadores o los miembros con mayor ascendencia tienen una mayor repercusión en la comunidad.
  Por ello, tanto las críticas como los elogios deben utilizarse cuidadosamente. Hay que recordar que las críticas continuas o injustificadas van a causar seguramente una mala reacción, al igual que los elogios. Sin embargo, en una discusión técnica, una crítica detallada puede considerarse en sí misma como algo positivo, ya que implica que la persona que la realiza se ha tomado el interés de analizar el diseño o la implementación objeto de la crítica.

- **Evitar la territorialidad**: una situación a evitar es aquella en la que ciertos miembros de la comunidad pretenden apropiarse de una parte ("su parte") del proyecto y no aceptan ninguna crítica o contribución de los otros miembros. Aunque al principio dicha actitud puede parecer positiva, ya que tales miembros suelen ser expertos y dedican mucho tiempo a su parte del proyecto, la consecuencia a largo plazo es que ningún otro desarrollador revisa el código realizado, con la consiguiente pérdida de calidad y la fragmentación de la comunidad.

- **Automatizar tareas**: en general, la mayoría de desarrolladores trabaja en una parte del código y desconoce qué es lo que hacen los demás. Por tanto, es responsabilidad de los coordinadores tener una visión global del proyecto y saber a qué se dedica cada miembro. Así resulta fácil identificar una serie de tareas inherentes al desarrollo del código que todos los desarrolladores llevan a cabo y que a menudo puede ser conveniente automatizar y centralizar.
  El ejemplo más evidente de esto es la automatización de tests[34], que permite que los desarrolladores realicen cambios y experimenten con partes del código con las que no están familiarizados.

- **Tratar adecuadamente a los usuarios**: la existencia de una comunidad de usuarios activa y que proporcione información valiosa a los desarrolladores es esencial para el éxito de todo proyecto de software libre. Sin embargo, a menudo los desarrolladores y los usuarios hablan, por así decirlo, lenguajes diferentes. Muchos usuarios no están familiarizados con el desarrollo de software ni con el funcionamiento de las comunidades de software libre. Los desarrolladores deben ser capaces de ponerse en el lugar de los usuarios y tratar de explicarse lo mejor posible.
  Hay que pensar que todo usuario puede contribuir a la comunidad en el futuro y, dado que la inmensa mayoría de usuarios no se dirige nunca a la comunidad de desarrolladores, hay que tratar con especial atención a los

(34)En particular, la creación de un paquete de pruebas, un programa que ejecuta el software del proyecto a fin de reproducir todos los errores conocidos y corregidos anteriormente. Esto permite que los desarrolladores se aseguren que no provocan de nuevo errores antiguos ya resueltos.

que sí lo hacen. Por ejemplo, cuando un usuario indica que una parte de la documentación está incompleta se le puede proponer que la complete él mismo y, cuando notifique un error, preguntarle si podría intentar resolverlo. Y, por supuesto, agradecer siempre su contribución, sea la que sea.

- **Compartir tareas de gestión**: además del desarrollo del código, todo proyecto lleva aparejadas una serie de tareas de gestión que, conforme el proyecto crece, se vuelven más complejas. Los coordinadores o el equipo que inició el proyecto suelen encargarse de ellas, pero es una buena práctica compartirlas con otros miembros del proyecto, tal y como se ha visto en el punto dedicado a la delegación. Entre estas tareas destacan:
  - **Gestión de parches**. Controlar qué parches se han recibido y analizarlos para aceptarlos o, en la mayoría de los casos, identificar sus problemas y notificarlos al autor del parche.
  - **Gestión de traducciones**. Coordinar la traducción de la documentación y del software.
  - **Gestión de documentación**. Mantener la documentación al día e integrar las modificaciones conforme éstas van apareciendo, así como la sección de preguntas frecuentes o FAQ.
  - **Gestión de errores**. Gestionar la base de datos de errores, lo que incluye, entre otras funciones, asegurar su integridad y evitar la existencia de errores duplicados.

- **Gestionar permisos**. una de las tareas de gestión que merece mención aparte es la gestión de los permisos o, en otras palabras, decidir quién tiene permiso para realizar *commits* y por tanto puede integrar su código en la copia remota del repositorio. Igualmente, la concesión de permisos implica también su posible revocación.
  Los desarrolladores que no tienen permiso para realizar *commits* pueden, por supuesto, contribuir al desarrollo del proyecto, realizando parches que resuelvan errores o añadan nuevas funcionalidades y que serán analizados por los desarrolladores del proyecto y finalmente incorporados. De hecho, el mecanismo más habitual de obtención del permiso para realizar *commits* es que un desarrollador contribuya con parches al proyecto, hasta que el equipo de desarrolladores considere que sus aportaciones y su conocimiento del proyecto son suficientemente valiosos.
  Para motivar la participación de nuevos desarrolladores, es conveniente que el procedimiento de obtención del permiso para realizar *commits* sea público y lo más transparente posible, así como su posible revocación.

3.5. *Releasing* y *packaging*

La preparación de las *releases* y el empaquetado o *packaging* son, junto con el desarrollo del código, unas de las tareas más importantes que deben realizarse en todo proyecto de software libre.

Una nueva *release* implica cambios, especialmente para los usuarios. En primer lugar, todos los errores conocidos de la anterior release han sido solucionados y muy probablemente haya otros nuevos. Además, es posible que haya funcionalidades y opciones de configuración nuevas. Incluso puede que aparezcan incompatibilidades entre la nueva versión del software y las anteriores, como por ejemplo en los formatos de los datos.

Ya que el cambio de una release a otra puede tener consecuencias importantes, y no todas positivas, uno de los primeros aspectos que se debe decidir es cómo se va a identificar cada una de las releases.

Para ello hay un buen número de convenciones más o menos creativas, pero lo más habitual es numerarlas con una serie de dígitos separados por puntos. Por ejemplo:

- Release 3.4.1
- Release 3.4.2
- Release 3.5
- Release 4.0

El significado de los dígitos puede variar. Los cambios en el tercer dígito suelen implicar soluciones a errores o pequeñas mejoras en algunas funcionalidades. Los cambios en el segundo dígito suelen significar la introducción de nuevas funcionalidades. Finalmente, los cambios en el primer dígito implican nuevos grupos de funcionalidades y, probablemente, cambios importantes en lo que respecta a la compatibilidad entre versiones.

Es conveniente indicar el significado de la numeración de las releases en el sitio web del proyecto.

Página web

Podéis consultar el esquema de versiones del proyecto APR (http://apr.apache.org/versioning.html).

Además, algunas *releases* se suelen identificar con la palabra *alpha* o *beta*, según su estado de desarrollo. Por ejemplo:

- Release 3.4.1 (alpha 1)
- Release 3.4.1 (alpha 2)
- Release 3.4.1 (beta)

En general, la palabra *alpha* se utiliza para designar la primera release, la cual permite que los usuarios accedan al software con todas las funcionalidades, pero en el que se espera aún un buen número de errores. Los usuarios que instalan y ejecutan una versión *alpha* suelen hacerlo para evaluar el software y notificar los errores al equipo de desarrolladores. Una versión *beta*, en cambio, está mucho más depurada y si casi no contiene errores se convertirá en la versión oficial: es lo que se llama una *versión candidata*.

Para los desarrolladores, un proyecto de software libre está continuamente en proceso de *release*, y utilizan siempre la última versión disponible en el repositorio para su desarrollo, por lo que puede ser difícil capturar el momento exacto de la *release*.

La mejor práctica para realizar esto es mantener una rama o *branch* en el repositorio que contenga el código que se introducirá en la próxima *release*, independientemente del tronco o *trunk*. De este modo, además, los desarrolladores que no están involucrados en la preparación de la *release* pueden seguir trabajando en el proyecto.

En consecuencia, una de las partes más importantes del proceso de preparación de una *release* es su estabilización, es decir, decidir qué cambios y funcionalidades van a integrarse en la rama de la próxima *release*. Aquí, el mecanismo de toma de decisiones de la comunidad de software libre debe ponerse de nuevo en marcha y otra vez hay dos alternativas principales:

- Designar un propietario de la *release*, o *release owner*, que decide qué cambios se van a introducir en la futura *release*.

- Votar los cambios que se van a introducir en la futura release, para lo cual es necesario definir las reglas de la votación. Una solución intermedia es fijar un número mínimo de desarrolladores que debe votar para que un determinado cambio se incluya.

Además, se puede nombrar uno o dos *release managers* encargados de integrar y validar los cambios en la rama de la *release*.

El software libre suele distribuirse como código fuente, adecuadamente empaquetado y comprimido en un formato estándar[35]. El nombre del paquete suele estar formado por el nombre del paquete, el número de versión y el sufijo apropiado según el formato. Por ejemplo:

- miproyecto-3.4.1.tar.gz
- miproyecto-3.4.2.zip

(35) En sistemas GNU/Linux, la convención es utilizar el formato TAR, comprimido por compress, gzip, bzip o bzip2. En sistemas Windows suele utilizarse el formato ZIP.

(36) Por ejemplo, el sistema RPM o DEB en sistemas GNU/Linux y, en sistemas Windows, los archivos MSI o ejecutables auto-instalables.

Entre la información que debe acompañar a toda nueva *release* se incluye la licencia bajo la cual se distribuye, las instrucciones de instalación y configuración y los cambios y las novedades respecto a la última *release*[36]. Esta información se recoge en una serie de archivos de número más o menos estándar: *LICENSE* o *COPYNG*, *README* o *INSTALL*, y *CHANGES*.

Finalmente, el usuario debe realizar la compilación del código fuente y su instalación en el sistema, que debe realizarse siempre de manera estándar si se desea que el software llegue al mayor número de usuarios posible. Otra posibilidad, empleada sobre todo con software ya maduro, es la distribución de paquetes binarios, ya sea como ejecutables o instalables, que en cualquier caso evitan que el usuario deba realizar el proceso de compilación manualmente[36].

Desde el punto de vista de la empresa de software libre, la política de *releases* es una de las herramientas más importantes para atraer a los usuarios potenciales del software. Una planificación adecuada de las *releases* debe dar respuesta puntualmente a las necesidades de los usuarios, ya sea en nuevas funcionalidades como en la corrección de errores, por lo que debe encontrarse el ritmo adecuado de publicación de las nuevas *releases*.

Por ejemplo, publicar nuevas *releases* con demasiada frecuencia puede saturar al usuario, que probablemente no las instalaría todas y, al contrario, dejar pasar demasiado tiempo entre *releases* podría hacer que el usuario buscara soluciones alternativas. Del mismo modo, es conveniente asegurar la calidad de las nuevas *releases*, intentando corregir la mayor cantidad de errores antes de su publicación. El efecto de una release llena de errores da una imagen muy mala del proyecto y de la empresa, que luego puede resultar difícil de enmendar. Por ello, es especialmente importante apoyarse en el carácter abierto y cooperativo del software libre para mejorar su calidad.

3.6. Elección de licencias

Las diferencias, las ventajas y los inconvenientes de cada una de las licencias de software libre es uno de los temas de discusión más recurrentes. Sin embargo, lo cierto es que la elección de una u otra licencia tiene un papel menor en la adopción y el éxito del proyecto, siempre y cuando ésta sea de software libre. La inmensa mayoría de los usuarios elige una determinada solución según la funcionalidad y la calidad que ofrece, pero no según su licencia.

Lo más importante es tener claro cuáles son los objetivos del proyecto y cuáles son los objetivos de la empresa de software libre respecto al proyecto y, en función de ello, elegir la licencia más adecuada o bien definir una nueva basada en las ya existentes[37].

(37)El anexo I proporciona una lista breve de las principales licencias utilizadas en la producción de software libre.

Así, numerosos proyectos de software libre proporcionan su propia licencia, adaptada a sus necesidades y objetivos[38].

(38)Por ejemplo, la licencia OpenBravo (http://www.openbravo.com/product/legal/license/) o la licencia dual de MySQL (http://www.mysql.com/about/legal/licensing/).

Las licencias de software libre, las relaciones y las incompatibilidades potenciales entre ellas pueden resultar muy complejas y, en ocasiones, puede ser necesaria la ayuda de abogados o juristas especializados.

Ved también

La asignatura del máster oficial de Software libre, *Aspectos legales y de explotación del software libre*, profundiza en estos aspectos.

Una de las principales fuentes de incompatibilidades es la reutilización de componentes libres bajo licencias restrictivas. Un ejemplo típico es la licencia GPL, que obliga a que todo software que utilice componentes GPL sea a su vez distribuido bajo licencia GPL.

Una buena práctica es la realización, desde el inicio del proyecto, de un inventario o mapa del software externo y las licencias utilizadas en el proyecto, que describa en qué partes del código se utilizan.

Resumen

En este módulo se han estudiado los fundamentos del negocio del software libre, poniendo de relieve la trascendencia que el modelo de software libre proporciona a la empresa tecnológica.

A grandes rasgos, los principales modelos de negocio del software libre se articulan en torno a la prestación de servicios relacionados con el propio software, puesto que normalmente este último se puede obtener libremente (aunque no se excluye la posibilidad de comprarlo o de poder adquirir variantes con funcionalidades adicionales). Servicios como los desarrollos a medida, la consultoría, la instalación y la integración de componentes, la migración de sistemas, la administración y el mantenimiento, el soporte y la formación son habituales en las empresas de software libre.

Asimismo, se han presentado, a grandes rasgos, las principales ideas de un plan de empresa basado en software libre, agrupando definiciones importantes para la posterior empresa, como la definición general, la misión y la visión, la producción, la organización interna y de recursos humanos, el estudio de mercado y el consecuente plan de marketing, los imprescindibles planes económicos y financieros, y la forma legal de la empresa. Y como el plan de empresa es también la planificación de un proyecto importante, es necesaria la gestión de los posibles riesgos que puedan aparecer.

Más adelante se trata la producción de software como un factor fundamental para la existencia empresarial, en especial, de la mayoría de los modelos de negocio vistos en este módulo. Aspectos importantes de la producción de software libre son las características que presentan en su creación, la infraestructura necesaria para su funcionamiento cotidiano, la organización de la comunidad de usuarios y desarrolladores (imprescindible en cualquier proyecto de software libre), la organización del desarrollo de código, la formalización del *releasing*, el *packaging* para la distribución y la definición de la imprescindible licencia libre.

En último lugar, el módulo dispone de dos anexos, el primero con una revisión breve de algunas de las licencias libres más comunes, como GNU/GPL, BSD License o Apache License, y el segundo con una relación de estándares y de organismos de estandarización que permiten el desarrollo de empresas de software libre.

Anexos

Anexo I: Licencias libres de software

A continuación se presenta una lista breve de las principales licencias utilizadas para la producción de software libre. Algunas de ellas se presentan con detalle en los materiales de las asignaturas de Introducción al software libre y de Aspectos legales y de explotación del software libre (2ª parte).

GNU/GPL v3

Las siglas GNU/GPL corresponden a la licencia *General Public License* del proyecto GNU.

Recurso web

Encontraréis más información sobre la licencia GNU/GPL en http://www.gnu.org/licenses/gpl.html.

Mantiene una política de redistribución robusta, llamada copyleft, que establece que todas las obras derivadas hereden la licencia original, incluso si se han combinado con otras. No se permite el enlace desde módulos con una licencia diferente. La política de protección de los derechos originales del autor y de la obra, entre otros, hace que la licencia GNU/GPL no sea compatible con cualquier otra licencia, como por ejemplo la licencia BSD original o las licencias propietarias.

Recurso web

Encontraréis una lista completa de compatibilidades en http://www.gnu.org/licenses/license-list.html.

La versión 3 de GNU/GPL no es directamente compatible con la versión 2. Sin embargo, muchos programas licenciados bajo la segunda versión permiten la utilización, en los mismos términos, de versiones posteriores de la licencia.

GNU/LGPL v3

Las siglas GNU/LGPL corresponden a la licencia *Lesser General Public License* del proyecto GNU, una licencia derivada de GNU/GPL.

Recurso web

Encontraréis más información sobre la licencia GNU/LGPL en http://www.gnu.org/licenses/lgpl.html.

Esta licencia se creó originalmente con el objetivo de permitir el uso, el enlace y la integración de bibliotecas y librerías de software libre con otros tipos de licencia, en ocasiones propietarias, salvando las restricciones de las licencias GNU/GPL. La práctica ha permitido licenciar un buen número de programas, algunos de amplia difusión en la actualidad.

Entre los programas licenciados bajo GNU/LGPL figura el paquete ofimático OpenOffice.org.

La versión 3 de GNU/LGPL no es directamente compatible con la versión 2. Sin embargo, muchos programas licenciados bajo la segunda versión permiten la utilización, en los mismos términos, de versiones posteriores de la licencia.

Licencia BSD

Las siglas BSD corresponden a la licencia Berkeley Software Distribution de la Universidad de Berkeley.

Recursos web

Encontraréis más información de la licencia BSD en http://www.debian.org/misc/bsd.license. También podéis encontrar un ejemplo de licencia derivada de la BSD original en http://www.freebsd.org/copyright/freebsd-license.html, llamada licencia BSD de dos cláusulas (*2-clause BSD license*) debido a la abolición de dos cláusulas de la licencia original.

Forma parte de un grupo de licencias (*BSD-style o BSD-like licenses*, entre ellas la licencia *FreeBSD*) denominadas permisivas, ya que mantienen una política poco restrictiva con los derechos del usuario. Esta política, llamada copycenter en oposición al término copyleft de las licencias GNU, permite entre otras cosas la aplicación comercial del producto, su conversión a código propietario y el enlace desde módulos con licencia diferente.

La licencia BSD original incorpora una cláusula de publicidad que la hace incompatible con GNU/GPL. La cláusula fue abolida en versiones posteriores, dando lugar a la llamada BSD de tres cláusulas (en inglés, *3-clause BSD license*), compatible con GNU/GPL.

MPL 1.1

Las siglas MPL corresponden a la licencia *Mozilla Public License* de la Fundación Mozilla (*Mozilla Foundation*).

Página web

Encontraréis más información sobre la licencia MPL en http://www.mozilla.org/MPL/MPL-1.1.html.

Surgió de la iniciativa privada y es un híbrido entre las licencias BSD y GNU/GPL. Está considerada una licencia permisiva semi-copyleft porque mantiene alguna posibilidad de establecer licencias propietarias a partir de obras derivadas. Permite el enlace desde módulos con licencia diferente. El artículo 13 permite licenciar una o más partes del código con una licencia diferente, llamada alternativa. Sólo en caso de que la licencia alternativa sea GNU/GPL -o cualquier otra de compatible-, la parte actuará como compatible con GPL, y se podrá enlazar con otros que también lo sean.

Licencia Apache 2.0

La licencia Apache es una licencia de la Fundación de Software Apache (*Apache Software Foundation*).

Página web

Encontraréis más información sobre la licencia Apache en http://www.apache.org/licenses/LICENSE-2.0.

Es bastante similar a la licencia BSD y está considerada permisiva porque mantiene la posibilidad de establecer licencias propietarias a partir de obras derivadas, así como enlazarla desde módulos con licencia diferente. El uso de pa-

tentes de esta licencia y las provisiones de indemnización la hacen compatible únicamente con la versión 3 de GNU/GPL, manteniendo la incompatibilidad con las dos versiones anteriores.

Licencia X11

La licencia X11, llamada erróneamente *MIT License*, es una licencia del Instituto Tecnológico de Massachussetts (*Massachussetts Institute of Technology, MIT*).

Página web

Encontraréis más información sobre la licencia X11 en http://www.opensource.org/licenses/mit-license.php.

Es bastante similar a la licencia BSD de tres cláusulas y está considerada permisiva porque permite licenciar obras derivadas como software propietario, así como enlazarla desde módulos con licencia diferente. Es compatible con GNU/GPL y está relacionada con el proyecto X.Org. En este sentido, algunas versiones antiguas de XFree86 la continúan utilizando, mientras que las versiones más modernas utilizan la licencia XFree86 1.1, que es incompatible con GNU/GPL debido a los reconocimientos que se imponen en toda la documentación.

Página web

Encontraréis más información sobre el proyecto X.Org en http://www.x.org/.

CDDL 1.0

Las siglas CDDL corresponden a la licencia *Common Development and Distribution License* de SUN Microsystems.

Página web

Encontraréis más información sobre la licencia CDDL en http://www.sun.com/cddl.

Está basada en la versión 1.1 de la licencia MPL. Las diferencias principales se centran en dos aspectos:

- El autor (o el poseedor de los derechos de autoría) puede restringir la jurisdicción legal de los derechos y los deberes de los usuarios del software.- El autor (o el poseedor de los derechos de autoría) puede restringir la jurisdicción legal de los derechos y los deberes de los usuarios del software.

- La licencia establece el requisito de identificar a todos los autores que contribuyen a las modificaciones introducidas en las obras derivadas.

Permite el enlace desde módulos con otra licencia, así como licenciar los derivados con una licencia diferente, eventualmente propietaria. Las características relacionadas con la propiedad intelectual la hacen incompatible con GNU/GPL.

CPL 1.0

Las siglas CPL corresponden a la licencia *Common Public License* de IBM.

Página web

Encontraréis más información sobre la licencia CPL en http://www-128.ibm.com/developerworks/library/os-cpl.html.

Su objetivo es promocionar el desarrollo de código libre, manteniendo la posibilidad de combinar el código con otras licencias, incluyendo licencias propietarias, aunque impide licenciar los derivados con otro tipo de licencia. Tam-

bién prohíbe que el código derivado infrinja las patentes del original, ya que obliga al pago de cualquier *royalty*. Es incompatible con GNU/GPL debido a las cláusulas relacionadas con la legalidad de las obras derivadas.

És incompatible amb GNU/GPL degut a les clàusules relacionades amb la legalitat de les obres derivades.

EPL 1.0

Las siglas EPL corresponden a la licencia Eclipse Public License de la Fundación Eclipse (*Eclipse Foundation*).

Página web

Encontraréis más información sobre la licencia EPL en http://www.eclipse.org/org/documents/epl-v10.php.

Se basa en la licencia CPL y mantiene una política permisiva orientada al negocio. La principal diferencia con respecto a CPL está en el tratamiento de las infracciones de patentes por parte de los contribuyentes al software. Todo el código licenciado bajo EPL mantiene la licencia en las obras derivadas. Sin embargo, permite licenciar las adendas separadamente bajo otros tipos de licencia, eventualmente propietarias. Permite el enlace desde módulos con licencia diferente. Las características relacionadas con la permisividad de la obra derivada y la gestión de la autoría la hacen incompatible con GNU/GPL.

Anexo II: Estándares abiertos

Definición

El proyecto SELF define un estándar abierto como un formato o protocolo que:

- está sujeto a la utilización y la evaluación pública, sin apremios, y de forma equitativa para todo el mundo;
- no tiene ningún componente o extensión que dependa de formatos o protocolos que no cumplan esta definición de estándar abierto;
- no tiene cláusulas técnicas o legales que limiten su utilización por cualquiera de las partes o en cualquier modelo de negocio;
- ha sido gestionado y desarrollado independientemente de los intereses comerciales particulares y mediante un proceso abierto y equitativo entre competidores y terceras partes; y
- se encuentra disponible en múltiples implementaciones de diferentes vendedores, o bien en una sola implementación disponible equitativamente para todas las partes.

Página web

Encontraréis más información sobre los estándares abiertos del proyecto SELF en http://selfproject.eu/en/OSD.html.

Sin embargo, no existe una definición única de estándar abierto, ya que cada organización preconiza un conjunto de características o prácticas adecuadas a sus objetivos particulares. Estas organizaciones pueden ser organizaciones de desarrollo de estándares, consejos supranacionales o gobiernos estatales. Algunas definiciones imponen la publicación en condiciones razonables y no discriminatorias (en inglés, *Reasonable And Non Discriminatory* (RAND)), es decir, no totalmente exentas de *royalties*.

Ejemplo

Como por ejemplo la definición de la Unión Europea (http://ec.europa.eu/idabc/servlets/Doca2cd.pdf) o del ITU-T (http://www.itu.int/ITU-T/othergroups/ipr-adhoc/openstandards.html).

Otras definiciones inciden más en las características del proceso que debe seguir un estándar para ser abierto, como por ejemplo las recomendaciones del World Wide Web Consortium (W3C), de Bruce Perens o de Ken Krechmer.

Recursos web

Encontraréis más información sobre estos procesos en:

http://www.w3.org/Consortium/Process. http://perens.com/OpenStandards/Definition.html. http://www.csrstds.com/openstds.pdf.

Organismos

Los principales organismos, asociaciones, institutos y consorcios relacionados con los estándares de las tecnologías de la información son los siguientes:

Página web

Encontraréis más información sobre ANSI en http://www.ansi.org/.

ANSI (*American National Standards Institute*): el Instituto Americano de Estándares Nacionales.

ETSI (*European Telecommunications Standards Institute*): el Instituto Europeo de Estándares de Telecomunicación (http://www.etsi.org/).

FreeStandards (*The Free Standards Group*): organización independiente que promueve la utilización y la aceptación de las tecnologías libres a través de estándares. (http://www.freestandards.org/.)

ICANN (*Internet Corporation for Assigned Names and Numbers*): Corporación de Internet para la Asignación de Nombres y Números (http://www.icann.org/).

IEC (*International Electrotechnical Commission*): Comisión Internacional de Electrotecnia (http://www.iec.ch/).

IEEE (*Institute of Electrical and Electronics Engineers, Inc*): Instituto de Ingenieros Eléctricos y Electrónicos (http://www.ieee.org/).

IETF (*Internet Engineering Task Force*): Grupo de Trabajo en Ingeniería de Internet (http://www.ietf.org/).

ISO (*International Organization for Standardization*): Organización Internacional para la Estandarización (http://www.iso.ch/).

ITU (*International Telecommunications Union*): la Unión Internacional de Telecomunicaciones agrupa entidades de los sectores público y privado para coordinar las telecomunicaciones y los servicios globales (http://www.itu.int/).

JXTA (*JXTA Project*): se trata de una combinación de estándares abiertos de igual a igual (en inglés, *peer-to-peer* o P2P) con implementaciones abiertas a Java. (http://jxta.kenai.com/)

OASIS (*Organization for the Advancement of Structured Information Standards*): es un consorcio internacional sin fines de lucro que orienta el desarrollo, la convergencia y la adopción de los estándares *e-business* (http://www.oasis-open.org/).

OpenGroup (*The Open Group*): es un consorcio internacional de vendedores para el avance neutral de la tecnología (http://www.opengroup.org/).

RossettaNet (*Open e-business process standards*): es una asociación para favorecer los estándares abiertos en el *e-business* (http://www.rosettanet.org/).

VoiceXML (*Voice XML Forum*): es una organización de industrias para crear y promover el *Voice Extensible Markup Language* (VoiceXML) (http://www.voicexml.org/).

W3C (*World Wide Web Consortium*): es un consorcio mundial para promover los estándares en Internet (http://www.voicexml.org/).

WS-I (*Web Services Interoperability Organization*): Organización para la Interoperabilidad de Servicios Web (http://www.ws-i.org/).

Estándares abiertos

A continuación se enumeran los principales estándares abiertos identificados por el proyecto SELF (http://selfproject.eu/standardshtml.html).

- **Texto no formateado**
  ASCII, ISO8859 (http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=28245) y UNICODE (http://www.unicode.org/).

- **Texto con formato**
  ODT (*Open Document Text*) y DocBook (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office).

- **Texto científico**
  ODF (*Open Document Formulae*), MathML (*Mathematical Markup Language*) (http://www.w3.org/Math/) y TeX/LaTeX (http://www.tug.org/) y (http://www.latex-project.org/).

- **Imágenes (tramas)**
  JPEG (*Joint Photographic Expert Group*) (http://www.jpeg.org/) y (http://www.w3.org/Graphics/JPEG/), PNG (*Portable Network Graphics*) (http://

www.libpng.org/pub/png/) y (http://www.w3.org/Graphics/PNG/), PNM (*Portable Any Map*) (http://netpbm.sourceforge.net/doc/pnm.html), GIF (*Graphics Interchange Format*) (http://www.w3.org/Graphics/GIF/spec-gif89a.txt), BMP (*Bitmap*) (http://atlc.sourceforge.net/bmp.html).

- **Imágenes (vectores)**
  SVG (*Scalable Vector Graphics*) (http://svg.org/) y (http://www.w3.org/Graphics/SVG/), ODG (*Open Document Graphics*).

- **Video**
  OpenEXR (http://www.openexr.com/), Theora (http://theora.org/), RIFF (*Resource Interchange File Format*) (http://msdn2.microsoft.com/en-us/library/ms713231.aspx), AVI (*Audio Video Interleave*) (http://msdn2.microsoft.com/en-us/library/ms779636.aspx).

- **Impresión**
  PDF (*Portable Document Format*) (http://www.adobe.com/devnet/pdf/), PS (*PostScript*) (http://partners.adobe.com/public/developer/ps/index_specs.html).

- **Hipertexto**
  HTML (*Hyper Text Markup Language*), XHTML (*Extended Hyper Text Markup Language*) (http://www.w3.org/MarkUp/).

- **Presentación**
  ODP (*Open Document Presentation*).

- **Audio**
  Vorbis (*OGG Vorbis*) (http://www.vorbis.com/) y (http://xiph.org/), FLAC (*Free LossLess Audio Codec*) (http://flac.sourceforge.net/)) y (http://xiph.org/), RIFF, WAV (Wave) (http://www.borg.com/~jglatt/tech/wave.htm).

- **Educació i aprenentatge**
  LOM (*Learning Object Metadata*) (http://zope.cetis.ac.uk/profiles/uklomcore), SCORM (*Sharable Content Object Reference Model*) (http://www.conform2scorm.com/), IMS (http://www.imsglobal.org/commoncartridge.html), LD (*Learning Design*) (http://www.imsglobal.org/learningdesign/index.html).

- **Empresa**
  XBRL (*Extensible Business Reporting Language*) (http://www.xbrl.org/).