

LICENCIAS DE SOFTWARE

- Qué es una licencia de SW?
- Contrato entre el titular del derecho de autor y el usuario del programa informático para utilizar éste en una forma determinada y de conformidad con condiciones convenidas

-73-

CLASIFICACIÓN

- Licencia de software libre
 - sin protección heredada
 - con protección heredada
- Licencia de código abierto
- Licencia de software propietario
 - Shareware o Freeware están también en esta categoría

-74-

¿QUÉ ES EL SOFTWARE LIBRE?

- Las cuatro libertades del software libre:
- **Libertad 00:** la libertad para **ejecutar el programa** con cualquier fin.
- **Libertad 01:** la libertad para **estudiar y modificar el programa**.
- **Libertad 10:** la libertad de **copiar el programa** de manera que puedas ayudar a tus pares.
- **Libertad 11:** la libertad de **mejorar el programa** y de **hacer públicas esas mejoras**, para beneficio de toda la comunidad.

-75-

LICENCIAS DE SOFTWARE LIBRE

- **Licencia Pública General de GNU (GNU GPL):** el autor conserva los derechos de autor (copyright), y permite la redistribución y modificación bajo términos diseñados
- Permite asegurar que todas las versiones modificadas del software permanecen bajo GNU GPL. No permite que se creen productos propietarios a partir de productos GPL
- Se usa en un 60% del software libre

-76-

LICENCIAS DE SOFTWARE LIBRE

- La GPL divide a las licencias de software libre se dividan en dos grandes grupos: aquellas que pueden ser mezcladas con código licenciado bajo GNU GPL y las que no lo permiten, por ejemplo:
 - **Las licencias de tipo BSD:** se utilizan en gran cantidad de software distribuido junto a los sistemas operativos BSD

-77-

LICENCIAS DE SOFTWARE LIBRE

- BSD mantiene la protección de copyright únicamente para requerir la adecuada atribución de la autoría en trabajos derivados
- Permite la libre redistribución y modificación, incluso si dichos trabajos tienen propietario
- Esto provoca que sea sensible al EEE (embrace, extend and extinguish) [ver mas](#)

-78-

LICENCIAS OPEN SOURCE

- Open Source es filosóficamente diferente del movimiento del software libre.
- Apareció en 1998 con Eric S. Raymond y Bruce Perens como principales contribuyentes fundando la OSI.
- **Objetivo:**
 - darle mayor relevancia a los beneficios prácticos del compartir el código fuente
 - Interesar a la industria en este concepto

-79-

LICENCIAS OPEN SOURCE

- La OSI sólo aprueba las licencias que se ajustan a la OSD (Open Source Definition)
- Cuando los programadores en internet pueden leer, modificar y redistribuir el código fuente de un programa, éste evoluciona y mejora.
- Los usuarios lo adaptan a sus necesidades, corrigen sus errores a una velocidad impresionante, mayor a la aplicada en el desarrollo de software convencional o cerrado

-80-

PREMISAS DEL MOVIMIENTO OS

- 1) Libre redistribución: el software debe poder ser regalado o vendido libremente.
- 2) Código fuente: el código fuente debe estar incluido u obtenerse libremente.
- 3) Trabajos derivados: la redistribución de modificaciones debe estar permitida.
- 4) Integridad del código fuente del autor: las licencias pueden requerir que las modificaciones sean redistribuidas sólo como parches.

-81-

PREMISAS DEL MOVIMIENTO OS

- 5) Sin discriminación de personas o grupos: nadie puede dejarse fuera.
- 6) Sin discriminación de áreas de iniciativa: los usuarios comerciales no pueden ser excluidos.
- 7) Distribución de la licencia: deben aplicarse los mismos derechos a todo el que reciba el programa
- 8) La licencia no debe ser específica de un producto: el programa no puede licenciarse solo como parte de una distribución mayor.

-82-

PREMISAS DEL MOVIMIENTO OS

- 9) La licencia no debe restringir otro software: la licencia no puede obligar a que algún otro software que sea distribuido con el software abierto deba también ser de código abierto.
- 10) La licencia debe ser tecnológicamente neutral: no debe requerirse la aceptación de la licencia por medio de un acceso por click de ratón o de otra forma específica del medio de soporte del software.

-83-

SOFTWARE PROPIETARIO

- Los propietarios establecen los derechos de uso, distribución, redistribución, copia, modificación y cesión.
- No permiten que el software sea modificado, desensamblado, copiado o distribuido de forma ilegal (piratería de software).
- Regula el número de copias que pueden ser instaladas e incluso los fines concretos para los cuales puede ser utilizado
- Limitan fuertemente la responsabilidad derivada de fallos en el programa.

-84-

SOFTWARE PROPIETARIO

- Los fabricantes de programas sometidos a este tipo de licencias por lo general ofrecen servicios de soporte técnico y actualizaciones durante el tiempo de vida del producto.
- Algunos ejemplos de este tipo de licencias son las llamadas CLUFs: Contrato de Licencia para Usuario Final o EULAs: End User License Agreement, por sus siglas en Inglés.

-85-

Modelos de comercialización del Software

86

DOS MODELOS... ¿ANTAGÓNICOS?

- ¿Qué **producto** vende la compañía de cable, o mejor aún Wal-Mart?
- ¿Qué **servicio** vende un fabricante de tuercas, o un productor de soja?
- Pero, ¿el **software**, en qué categoría debería estar?
 - ¡puede ser clasificado en **ambas!**

-87-

CARACTERÍSTICAS DE LOS PRODUCTOS

- La fabricación de productos tiene ciertas características en común:
- Se requiere contar con un **capital previo**.
- Primero se **diseña** el producto. Luego se **replica** ininidad de veces.
- Más tarde se **reparte** a través de alguna cadena de distribución.
- Finalmente, el cliente **paga por recibir una copia** del producto. El producto suele ser una "cosa" tangible.

-88-

CARACTERÍSTICAS DE LOS SERVICIOS

- El proveer servicios también presenta ciertas características en común:
- No se requiere contar con un capital previo.
- Se debe encontrar **una necesidad insatisfecha** que tengan los clientes.
- No se requiere una cadena de distribución.
- El cliente **paga por tener su necesidad debidamente atendida**. No suele ser una “cosa” tangible.

-89-

EL SOFTWARE COMO PRODUCTO

- Muchas compañías de software exitosas adoptan este modelo
- Recordemos que el software en sus inicios **venía gratis con el hardware**.
- Recién a fines de los 70' principios de los 80' se “inventó” esta nueva forma de concebir al software.

-90-

EL SOFTWARE COMO PRODUCTO

- El software atraviesa las etapas propias de la fabricación de productos:
 - Es **diseñado** (aplicando técnica de ingeniería de software).
 - Luego es **replicado**.
 - Es eventualmente **distribuido**.
 - Y finalmente **vendido** (Wal-Mart's, Electronic Boutiques, CompuWorlds, etc.).
- Se genera una “cosa” tangible: **la cajita**.

-91-

VENTAJAS DEL SOFTWARE COMO PRODUCTO

- Esta concepción brinda varias ventajas:
- El costo de diseñar el software **se paga una vez y se cobra miles o millones de veces**.
- Podemos **elegir la licencia** que nos venga en gana, o bien diseñar nuestra propia licencia.
 - Ej: Prohibir que se pueda hablar mal del producto.

-92-

VENTAJAS DEL SOFTWARE COMO PRODUCTO

- Me permite agregar “**planned obsolescence**” para mantener el mercado cautivo.
 - Ej: Office 2003 no soporta OpenXML.
- Si logramos una **posición monopólica** en el mercado podremos **reducir los costos de desarrollo**.

-93-

VENTAJAS DEL SOFTWARE COMO PRODUCTO

- Más aun, podemos proteger nuestro monopolio mediante **patentes de software** que impidan o desalienten la competencia.
- Incluso podemos hacer plata litigando.

-94-

DESVENTAJAS DEL SOFTWARE COMO PRODUCTO

- No obstante, también existen inconvenientes:
- Es difícil contemplar las **necesidades** de miles o millones de usuarios a la vez.
- La **competencia** nos puede arruinar el negocio brindando un producto similar a menor costo.
- Aun sin competencia, **seguimos compitiendo con nosotros mismos**. El software no se estropea.

-95-

DESVENTAJAS DEL SOFTWARE COMO PRODUCTO

- El **soporte técnico** del producto vendido genera pérdidas.
 - **Solución:** dejar de soportar productos “legacy”.
- Nos transformamos en los **responsables legales** del funcionamiento del producto.
- La **piratería** nos afecta seriamente.

-96-

ASPECTOS NO TAN ÉTICOS DEL SOFTWARE COMO PRODUCTO

- Se supone que la ley de oferta y demanda asegura a los clientes obtener el **mejor producto** al **menor precio**.
- Si bien la competencia es buena para el **cliente**, no lo es para el **productor**.
- **Solución:** si logramos monopolizar el mercado, podemos evitar la competencia.
 - Ej: El juicio anti-trust a Microsoft.

-97-

PATENTES DE SOFTWARE

- ¿Qué sucedería si alguien registra que tuvo la idea original de que al final del cuento el asesino fue el mayordomo?
- ¿Qué son las **patentes de software**?
 - Las patentes de software son similares a las patentes de invención comunes, con la diferencia que en vez de patentar una cosa, estamos patentando una idea.
- Lamentablemente, esto pasa hoy en día...

-98-

LAS DOS CARAS DE LAS PATENTES DE SOFTWARE

- Son un **capital** para quien las posee:
 - Puedo impedir que la competencia provea un producto similar al mío. Ej: las ventas One-Click en Amazon vs. ebay.
 - Patentar una idea cuesta arriba de **50.000 USD**.
- Son un **riesgo** para el resto del mundo:
¿Qué pasa si mi programa incluye sin saberlo un algoritmo patentado?

-99-

EJEMPLOS DE PATENTES DE SOFTWARE

- Algunas patentes de software son bastante **ridículas**:
 - Microsoft patentó el operador que permite determinar si dos variables son diferentes.
 - Amazon patentó las ventas "one-click".
 - Adobe patentó mostrar una paleta de colores usando pestañas.

-100-

EJEMPLOS DE PATENTES DE SOFTWARE

- Algunas patentes de software son bastante **ridículas**:
 - Sun patentó como transformar los nombres de los archivos de W95 a NT.
 - IBM patentó reorganizar las ventanas cuando no se puede ver todo su contenido.

-101-

SISTEMAS DE PROTECCIÓN DEL SOFTWARE

- Existen dos modelos para los sistemas de protección del software:
 - Basados en **software**:
 - Siempre pueden ser crackeados.
 - Puede traer problemas de confiabilidad.

-102-

SISTEMAS DE PROTECCIÓN DEL SOFTWARE

- Basados en **hardware**:
 - Son apenas más seguros.
 - Molestan al usuario.
 - Pueden traer problemas de compatibilidad.

-103-

SOFTWARE COMO PRODUCTO

- En síntesis:
 - Este enfoque permite recaudar cuantiosas cantidades de dinero a las empresas ya establecidas (especialmente, las que controlan monopólicamente al mercado).
 - Es complicado que una empresa recién formada pueda tener éxito siguiendo este enfoque por diversas razones: Alta inversión inicial, Peligro de litigación, etc

-104-

EL SOFTWARE COMO SERVICIO

- El software también presenta ciertas características que lo hacen un servicio:
 - El software no se fabrica, se desarrolla.
- En vez de llevar el producto a los clientes, éstos vienen a pedirlo.
- Generan una “cosa” no tangible: el servicio que es brindado por el software.
Ej: Gmail, Amazon, eBay, Salesforce.

-105-

VENTAJAS DEL SOFTWARE COMO SERVICIO

- La mayoría del software sigue siendo **a medida**.
- El **mantenimiento** y el **soporte** del software insume muchos recursos:
 - El **60%** de todo el dinero gastado en el mundo cada año en software se va en mantenimiento.
 - El mantenimiento insume el **70-80%** del costo total del software.

-106-

VENTAJAS DEL SOFTWARE COMO SERVICIO

- Genera gran número de **puestos de trabajo**.
- Los usuarios en general **no están conformes** con el software que usan, siempre **desean adaptarlo** a sus necesidades.
 - Gran oportunidad de negocio.
 - Difícil de lograr sin acceso al código fuente.
 - Los plug-ins son una solución de compromiso.

-107-

VENTAJAS DEL SOFTWARE COMO SERVICIO

- El desarrollo que ha tenido internet brinda un nuevo espacio para el software como servicio: los **servicios web**.
 - El servicio que brindemos puede depender de servicios brindados a su vez por terceros.
- Gran oportunidad para aprovechar:
Reconversión de los sistemas existentes.

-108-

VENTAJAS DEL SOFTWARE COMO SERVICIO

- Grandes empresas que apuestan por esta nueva visión ya nos proveen de las **herramientas** necesarias, por caso:
 - La arquitectura .NET de Microsoft.
 - Los applets y los servlets de Sun.
 - La tecnología AJAX (Gmail).
- Se abaratan los costos de "**deployment**"
Actualizaciones continuas y transparentes para el usuario final.

-109-

VENTAJAS DEL SOFTWARE COMO SERVICIO

- Podemos usar los **repositorios de software libre** como punto de partida de nuestros desarrollos.
- O bien, podemos **usar software libre** como la infraestructura de nuestro servicio. Ej: La arquitectura **LAMP**.
- Por último, también podemos centrar nuestro servicio en torno al **soporte**.

-110-

¿QUÉ COMPRAMOS?

- Las compañías de software **no suelen vender software**.
 - Venden "**permisos**" para usar software.
 - Al comprar software enlatado estamos comprando una **licencia de uso**.

-111-

¿EN QUÉ CATEGORÍA CAE EL SOFTWARE LIBRE?

- El software libre **puede ser gratis o pago**.
- **Siempre es open source**.
 - Libre implica open source.
 - Open source no implica libre.
- Se encuentra protegido por **licencias del tipo copyleft**.
- **Siempre es legal**, de hecho sería imposible piratear software libre.

-112-

DESVENTAJAS DEL SOFTWARE COMO SERVICIO

- Algunos sostienen que se trata de la programación orientada a componentes bajo un nuevo disfraz.
 - El software como servicio implica **cambio**. El cambio siempre es resistido.

-113-

DESVENTAJAS DEL SOFTWARE COMO SERVICIO

- No ganamos nada con mantener una posición monopólica en el mercado.
- Las patentes de software constituyen un inconveniente, especialmente si basamos nuestro servicio en software libre o de código abierto.

-114-

SOFTWARE COMO SERVICIO

- En síntesis:
 - Parece brindar un cúmulo de alternativas que aun no han sido exploradas:
 - Software libre como punto de partida.
 - Software libre como infraestructura.
 - Servicios webs y la web semántica.

-115-

SOFTWARE COMO SERVICIO

- En síntesis:
 - Situación ideal para empresas recién formadas o en proceso de formación.
 - La principal desventaja es que es imposible mantener cautivo al mercado.
 - Se debe competir de forma honesta.

-116-

UN MODELO HÍBRIDO

- Finalmente, también es posible ensayar un modelo híbrido. Ej: el programa ["Software Assurance" de Microsoft](#).
- Por caso, podemos brindar el servicio de obtener de forma gratuita los productos que desarrollemos a lo largo de un cierto período.
- Goza y sufre de algunas de las ventajas y desventajas de ambos acercamiento.

-117-

CONCLUSIONES

- El software presenta características como **producto** y como **servicio**.
- Ambas concepciones tiene sus **ventajas** y **desventajas**.
- El software como servicio ha encontrado dos nuevos aliados:
 - La existencia de gran cantidad de **software libre de calidad**.
 - El interés en torno a los **servicios web**.

-118-

SALESFORCE

- Un ejemplo de software como servicio que hoy en día está generando ganancias considerables y lidera el mercado
- Ver la página web de la compañía

-119-

Mantenimiento de los SI

120

MANTENIMIENTO

- Se refieren al proceso de manejar los cambios a los sistemas de aplicación manteniendo la integridad del código fuente en producción y los ejecutables
- Un sistema en producción no se mantiene estático
- Es necesario un proceso para ordenar los cambios

-121-

PROCESO DE GESTIÓN DE CAMBIOS

- Comienza cuando se aprueba un cambio
- Debe existir una metodología para priorizar y aprobar cambios
- Son iniciados por distintos actores
- En caso de sistemas adquiridos el vendedor puede distribuir parches

-122-

PROCESO DE GESTIÓN DE CAMBIOS

- Los parches deben ser revisados por la gerencia de usuarios y del sistema
- Los usuarios deben proponer cambios usando algún tipo de comunicación formal
- Se debe poder hacer un seguimiento de los cambios

-123-

PROCESO DE GESTIÓN DE CAMBIOS

- Para eso se asigna un número a cada requerimiento y se ingresa en un sistema automático (ideal)
- Ver ejemplo de pag 212 del manual de CISA
- Toda la información relacionada con los cambios la debe mantener el staff encargado del mantenimiento del SI

-124-

PROCESO DE GESTIÓN DE CAMBIOS

- Deben existir registros del mantenimiento de los programas ya sea manuales o automáticos
- Mínimamente: ID del programador, fecha y hora, nro de requerimiento asociado, cambio en las líneas de código

-125-

SI HAY POCO PERSONAL

- Cuando la persona que crea el programa es también el operador se debe tener especial cuidado
- Se requieren controles compensatorios por la falta de separación de obligaciones
- Se requiere que se aprueben los cambios por la gerencia de usuarios antes de ponerlos en producción
- La gerencia puede tener un SW de control de cambios para automatizar esta tarea

-126-

SEPARACIÓN DE OBLIGACIONES

- Los programadores no deben tener acceso para modificar, escribir o borrar datos en producción
- Puede ser que ni siquiera tengan permiso de lectura sobre los datos almacenados.

-127-

APROBACIÓN

- Después de que el usuario final esté conforme con los resultados del testeo, se debe obtener la aprobación de la gerencia de usuarios
- Debe estar documentado en el RFC o en otro documento a tal fin
- Es importante que el staff de mantenimiento almacene esta evidencia

-128-

A TENER EN CUENTA

- Debe actualizarse la documentación existente para reflejar el cambio
- La documentación que se guarda offsite para recuperación de desastre debe ser actualizada también

-129-

TESTING DE LOS CAMBIOS

- Los programas cambiados deben ser testeados con la misma disciplina que los nuevos para asegurarse de que los cambios funcionan según lo pretendido
- Si el análisis de riesgo así lo indica se debe realizar testing adicional para asegurar los siguientes puntos

-130-

TESTEAR

- La funcionalidad existente no fue dañada por el cambio
- La performance no se degrada
- No se crearon exposiciones en cuanto a la seguridad

-131-

AUDITAR EL PROCESO DE CAMBIO

- Se debe asegurar que se proteja a los programas en producción de cambios no autorizados
- Para esto se debe cumplir con los siguientes objetivos de control

-132-

OBJETIVOS

- Restringir el acceso a las librerías
- Realizar supervisiones
- RFC aprobados y documentados
- Un formulario estándar para RFC
 - Especificación del cambio, análisis del costo, fecha estimada, firmado por quien lo propone y por quien lo autoriza, asignado a un equipo

-133-

OBJETIVOS

- Se debe seleccionar un subconjunto de cambios y revisar el proceso
- Si un grupo independiente actualiza los programas en producción se debe verificar si existen procedimientos para asegurar que se posee el RFC

-134-

CAMBIOS DE EMERGENCIA

- A veces se requieren cambios urgentes para que el SI siga operando
- En los manuales de operaciones debe estar especificado que hacer en esas situaciones
- En general se usan IDs especiales que permiten que el programador tenga acceso al ambiente de producción

-135-

CAMBIOS DE EMERGENCIA

- El uso de estas IDs debe ser logueado y monitoreado
- Los cambios de emergencia debe aplicar los procesos de manejo de cambio de forma retroactiva
- Se mantienen en una librería de emergencia hasta que se complete el proceso normal

-136-

DEPLOYMENT DE LOS CAMBIOS

- Una vez que la gerencia de usuarios aprueba los cambios los programas modificados se pueden poner en producción
- Un grupo independiente debería realizar esta función, por ejemplo QA
- Se deben implementar restricciones de acceso a través del SO o algún paquete externo

-137-

DEPLOYMENT DE LOS CAMBIOS

- En caso de SD cambiar todos los nodos es más complicado, se debe realizar con tiempo para asegurar que:
 - Se realicen controles sobre la conversión de los datos
 - Se entrene al personal
 - Se reduzca el riesgo de cambiar todos los nodos a la vez

-138-

CAMBIOS NO AUTORIZADOS (POR QUÉ SUCEDEN)

- El programador tiene acceso a las librerías en producción
- El usuario responsable de la aplicación no sabe del cambio
- No están establecidos los RFC estándares ni los procedimientos
- No firmaron para autorizar la realización del cambio

-139-

CAMBIOS NO AUTORIZADOS (POR QUÉ SUCEDEN)

- El responsable no autorizó que el cambio fuese introducido al sistema en producción
- El programador puso código extra para beneficio personal
- Los parches enviados no fueron testeados o el vendedor tiene acceso a cargar los cambios en el sistema en producción

-140-

REFERENCIAS

- Auditors guide to information systems auditing, R. Cascarino, Willey, 2007. Capítulo 18 y 20.
- Manual CISA 2013. Capítulo 3.
- VER <http://www.evaluandoerp.com/nota-198-Guia-para-escribir-un-RFP-%28request-for-proposal%29-.html>

-141-

