

Software Libre

Eduardo Grosclaude

2013-07-01

[V0.6 - Material en preparación, se ruega no imprimir mientras aparezca esta nota]

Resumen

En este escrito se presenta la descripción y material inicial de la asignatura **Software Libre**, para la carrera de Tecnicatura Universitaria en Administración de Sistemas y Software Libre, de la Universidad Nacional del Comahue.

La materia es cuatrimestral en modalidad presencial y las clases son de carácter teórico-práctico, desarrolladas en forma colaborativa. Está preparada con los objetivos generales de **conocer los aspectos técnicos, legales, económicos y sociales que distinguen al Software Libre y de Código Abierto; conocer las formas de analizar, evaluar y utilizar las fuentes de documentación y soporte del Software Libre y de Código Abierto.**

Página en blanco

Índice

I	La asignatura	6
1.	Objetivos	6
	De la carrera	6
	De la asignatura	6
2.	Cursado	6
3.	Contenidos	6
	Contenidos mínimos	6
	Programa	7
	Bibliografía inicial	7
4.	Evaluación	7
	Trabajo I - Colaboración con proyectos libres	7
	Etapa 1	7
	Etapa 2	8
	Etapa 3	8
	Trabajo II - Evaluación de proyectos libres	8
	Etapa 1	8
	Etapa 2	8
	Etapa 3	9
	Etapa 4	9
	Cronograma de ejecución	9
II	El Software Libre	10
1.	Software Libre y Código Abierto	10
	Aspectos éticos	10
	Proyectos Libres	11
	Preguntas	12
2.	Aspectos técnicos	13
	Código fuente y ejecutables	13
	Ciclo de compilación	13
	Proyectos de Software Libre	17
	La Catedral y el Bazar	17
	Infraestructura tecnológica	17
	Preguntas	17
3.	Aspectos legales	21
	Propiedad intelectual	21
	Licencias	21
	Licencias de software	22
	Licencias FSF	22
	GPLv1	22
	LGPL	22
	GPLv2	22
	GPLv3	22
	Aferro GPL o AGPL	22
	Licencias tipo BSD	23
	Licencias de contenidos libres	23

Preguntas	24
4. Uso y aplicación de Software Libre	26
Selección de Software Libre	26
Administración de TI	26
CobiT	26
Migraciones en general	27
Formas de migración a SL	28
Facilitar migración futura	28
Costo Total de Propiedad, CTP	28
Metodologías de análisis de CTP	29
5. Producción de y con Software Libre	30
Adopción estratégica de Software Libre	30
Desarrollo de proyectos de Software	30
Proyecto de software tradicional vs. SL	30
Modelo de desarrollo Open Source	31
Modelos de negocio	31
Motivaciones para producir SL	31
SL en las organizaciones	32
Administración pública	32
Sector privado	32
III Anexos	33
A. Licencias libres y no libres	33
Comparación de GPL y MS EULA	33
Características básicas de la EULA	33
Características básicas de la licencia libre GPL	33
B. Política de uso de SL en la UNC	33
Establecen la política de uso de Software Libre en la UNCo	33
C. Cuestionario para identificación de riesgos operativos	34
Taxonomía de Riesgos	34
A - Misión	34
B - Procesos	35
C - Restricciones	35

Página en blanco

Parte I

La asignatura

1. Objetivos

De la carrera

Según el documento fundamental de la Tecnicatura, el Técnico Superior en Administración de Sistemas y Software Libre estará capacitado para:

- Desarrollar actividades de administración de infraestructura. Comprendiendo la administración de sistemas, redes y los distintos componentes que forman la infraestructura de tecnología de una institución, ya sea pública o privada.
- Aportar criterios básicos para la toma de decisiones relativas a la adopción de nuevas tecnologías libres.
- Desempeñarse como soporte técnico, solucionando problemas afines por medio de la comunicación con comunidades de Software Libre, empresas y desarrolladores de software.
- Realizar tareas de trabajo en modo colaborativo, intrínseco al uso de tecnologías libres.
- Comprender y adoptar el estado del arte local, nacional y regional en lo referente a implementación de tecnologías libres. Tanto en los aspectos técnicos como legales.

De la asignatura

- Conocer los aspectos técnicos, legales, económicos y sociales que distinguen al Software Libre y de Código Abierto
- Conocer las formas de analizar, evaluar y utilizar las fuentes de documentación y soporte del Software Libre y de Código Abierto

2. Cursado

- Cuatrimestral de 16 semanas, 64 horas totales
- Clases teórico-prácticas presenciales
- Promocionable con trabajos prácticos

3. Contenidos

Contenidos mínimos

- Las licencias de software. Software Libre y Open Source. Comparación.
- Ventajas de la disponibilidad del código fuente.
- Modelos de desarrollo abiertos y colaborativos.
- Aspectos legales y de explotación del Software Libre.
- Implantación de sistemas de Software Libre. Factibilidad.
- Aspectos económicos y modelos de negocio del Software Libre.
- Costo total de operación. Comparación con otras alternativas.
- El Software Libre en el sector público, en la educación y en la empresa.

Programa

1. Introducción. Software Libre y Código Abierto. Aspectos éticos. Implicancias sociales. Localización. Proyectos libres.
2. Aspectos técnicos. Proyectos de Software Libre. Modelo de desarrollo. Infraestructura tecnológica. Manejo de documentación y soporte. Seguridad.
3. Aspectos legales. Dominio Público, Copyright, Copyleft y Licenciamiento. Licencias de FSF, Creative Commons, OSI, otras.
4. Uso y aplicación de Software Libre. Costo total de operación. Estudio de costo/beneficio. Procesos de migración.
5. Producción de Software Libre y con Software Libre. Modelos de negocio. Colaboración en proyectos. Organizaciones y software. Administración pública, educación pública, sector privado.

Bibliografía inicial

- **Introducción al Software Libre**, Jesús González Barahona, Joaquín Seoane Pascual y Gregorio Robles
- **Aspectos legales y de explotación del software libre**, Malcom Bain, Manuel Gallego Rodríguez, Manuel Martínez Ribas y Judit Rius Sanjuán
- **Guía práctica sobre Software Libre, su selección y aplicación local en América Latina y el Caribe**, Fernando Da Rosa y Federico Heinz

4. Evaluación

La evaluación de la materia se realizará mediante trabajos grupales de investigación y desarrollo sobre proyectos de Software Libre, de la siguiente manera.

- Los estudiantes se dividirán en grupos de 2 a 5 personas.
- Los grupos desarrollarán trabajos prácticos en etapas que se distribuirán a lo largo de la materia.
- Cada grupo abrirá un diario, blog o wiki de acceso público en cualquier sitio disponible y publicará, mediante el Foro de la materia, la forma de acceder al diario para lectura. Los docentes y los demás estudiantes de la materia podrán acceder al diario del grupo para lectura. Todo cambio en la dirección o forma de acceso deberá ser informado mediante el Foro.
- El grupo irá aportando los resultados de cada etapa de los trabajos a su diario, y periódicamente comentará además en clase las experiencias surgidas durante la realización de los trabajos.
- El material publicado en el diario será reunido en un documento final que será entregado **en formato electrónico** al finalizar la materia. El documento indicará tema del trabajo, resumen, integrantes del grupo, desarrollo y conclusiones.
- El documento será acompañado por una presentación de no más de treinta minutos que será expuesta según el cronograma adjunto.
- La acreditación final tendrá en cuenta la calidad del material aportado al diario por el grupo, la calidad de los documentos finales de los trabajos, la presentación oral y la participación en clase ofreciendo la experiencia adquirida durante la realización de los trabajos.

Trabajo I - Colaboración con proyectos libres

Etapas

Descargar e instalar software ofrecido por un proyecto de Software Libre que esté en actividad (puede tratarse de un entorno de escritorio, un programa de sistema, programas de usuario final, una distribución completa, etc.). Familiarizarse con el software utilizándolo.

Etapas 2

Basándose en el conocimiento adquirido con el uso del software, colaborar de alguna forma con el proyecto que lo origina:

- traduciendo o localizando parte del software,
- generando documentación faltante,
- traduciendo parte de la documentación,
- detectando y denunciando errores en el software o en la documentación,
- aportando, modificando o corrigiendo código,
- aportando conocimiento a los usuarios del proyecto en blogs, salas de chat, bases de conocimiento, etc.

Puede abordarse cualquier cantidad manejable de proyectos. La colaboración debe consistir en alguna interacción positiva y completa con cada proyecto. El grupo incorporará al diario los reportes que acrediten esa interacción. Cuando no sea posible realizar o completar la interacción se indicarán las causas, y las acciones realizadas.

El aporte al proyecto debe efectuarse por los canales establecidos por el proyecto. Si se trata de documentación, respetar el formato utilizado; si es el reporte de un error, hacerlo por la vía preferida por el proyecto, etc.

Etapas 3

El grupo entregará un documento conteniendo la historia de las interacciones con cada proyecto, adjuntando las pruebas en anexos y ofrecerá una presentación.

Trabajo II - Evaluación de proyectos libres

Etapas 1

El grupo enunciará un determinado requerimiento concreto de software que puede ser presentado por un empleador. Algunos ejemplos posibles son:

- “un servidor de correo electrónico que maneje listas”,
- “una aplicación de control de asistencia para empleados”,
- “un sistema de edición de textos para traductores”,
- “un sistema de gestión de contenidos web que incluya workflow”,
- “un motor de juegos 2D para crear juegos que asistan en la enseñanza de matemática”,
- “un programa de simulación de ataques para evaluar postura de seguridad”,
- “un sistema de control de stock para zapaterías”,
- “una distribución de GNU/Linux para escuelas de arte”,
- “una distribución para sistemas empujados”, etc.

El grupo debe comprender el propósito del software requerido y debe contar con al menos un integrante con conocimiento razonable de la temática involucrada. El grupo escribirá una entrada en el diario consignando toda la información posible sobre los requerimientos.

Etapas 2

- El grupo n (en adelante “el proveedor”) tomará a su cargo el requerimiento del grupo $n + 1$ (en adelante “el cliente”), y se atenderá a dicha descripción para el resto del trabajo.
- El grupo proveedor buscará proyectos de SL que apunten a cubrir esos requerimientos y seleccionará al menos dos proyectos, idealmente tres, de entre ellos.

Etapa 3

Los proyectos serán comparados en función de varios parámetros o dimensiones.

- ajuste a los requerimientos (actual, previsto o potencial),
- licenciamiento,
- motivación del desarrollo,
- modelos de negocio del proyecto,
- tamaño y permanencia de la comunidad,
- dinámica de soporte,
- dinámica de actualizaciones y mejoras del software.

Se podrán agregar a la comparación uno o más desarrollos no libres.

Las dudas sobre detalles de los requerimientos serán dirigidas al grupo cliente, y contestadas por aquél, mediante el Foro de la página de la materia.

Etapa 4

El grupo entregará un documento conteniendo la comparación y haciendo una recomendación final, explicando sus fundamentos. Deberán volcar en el trabajo lo que se vaya aprendiendo durante el curso de la materia, en cada uno de los parámetros o dimensiones nombrados. Finalmente ofrecerán una presentación sobre el trabajo.

Cronograma de ejecución

Semana	Unidad	Trabajo I	Trabajo II
1 2	1. Introducción, Software Libre	Etapa 1	
3 4 5 6	2. Aspectos técnicos	Etapa 2	
7 8 9	3. Aspectos legales	Etapa 3 Entrega y presentaciones	Etapas 1 y 2
10 11 12 13	4. Uso de SL		Etapa 3
14 15 16	5. Producción de SL		Etapa 4 Entrega y presentaciones

Parte II

El Software Libre

1. Software Libre y Código Abierto

- Software Libre, Open Source/Código Abierto, FOSS o FLOSS
<http://drupal.usla.org.ar/page/%C2%BFque-es-el-software-libre>
Free \neq Gratis
- El proyecto GNU y la FSF
<http://www.fsfla.org>
http://es.wikipedia.org/wiki/Portal:Software_libre
- Las cuatro libertades
 0. De correr el software, con cualquier propósito
 1. De estudiar cómo está hecho, para poder adaptarlo a sus propias necesidades
 2. De copiarlo y darlo a otras personas para poder ayudarlas
 3. De mejorarlo y donar el resultado a la comunidad, para permitir el avance colectivo
- Código fuente, código objeto
- Open Source Initiative
<http://opensource.org/>
- The Open Source Definition - <http://www.opensource.org/docs/osd>
 1. Libre Redistribución
 2. Debe incluir el Código Fuente
 3. La licencia debe permitir Obras Derivadas
 4. Integridad de los fuentes del autor
 5. No Discriminación contra personas o grupos
 6. No Discriminación contra campos de actividad
 7. Los derechos asignados por la licencia deben alcanzar a todos los recipientes automáticamente
 8. La licencia no debe ser específica de un producto
 9. La licencia no debe restringir el uso de otro software
 10. La licencia debe ser tecnológicamente neutra

Aspectos éticos

- Valores
 - Cooperación más importante que individualidades
 - “Si los dos tenemos una manzana/una idea”
 - “Si no está hecho es porque no lo hiciste”
 - “Con suficientes ojos se detectan todos los errores”
 - Puedo aprovechar lo que me ofrecen libremente en lugar de *piratear*
 - Puedo comprender los mecanismos, trascender la herramienta y acceder a los conceptos
 - Puedo ser parte del desarrollo de la tecnología y cooperativamente contribuir al mejoramiento
- Implicancias sociales
 - Equilibrar la balanza de pagos internacional del conocimiento
 - Usar un producto propietario tributa fondos al proveedor
 - Usar proyectos libres aplica los fondos a quienes ofrecen los servicios
 - Crítica al concepto de Propiedad Intelectual

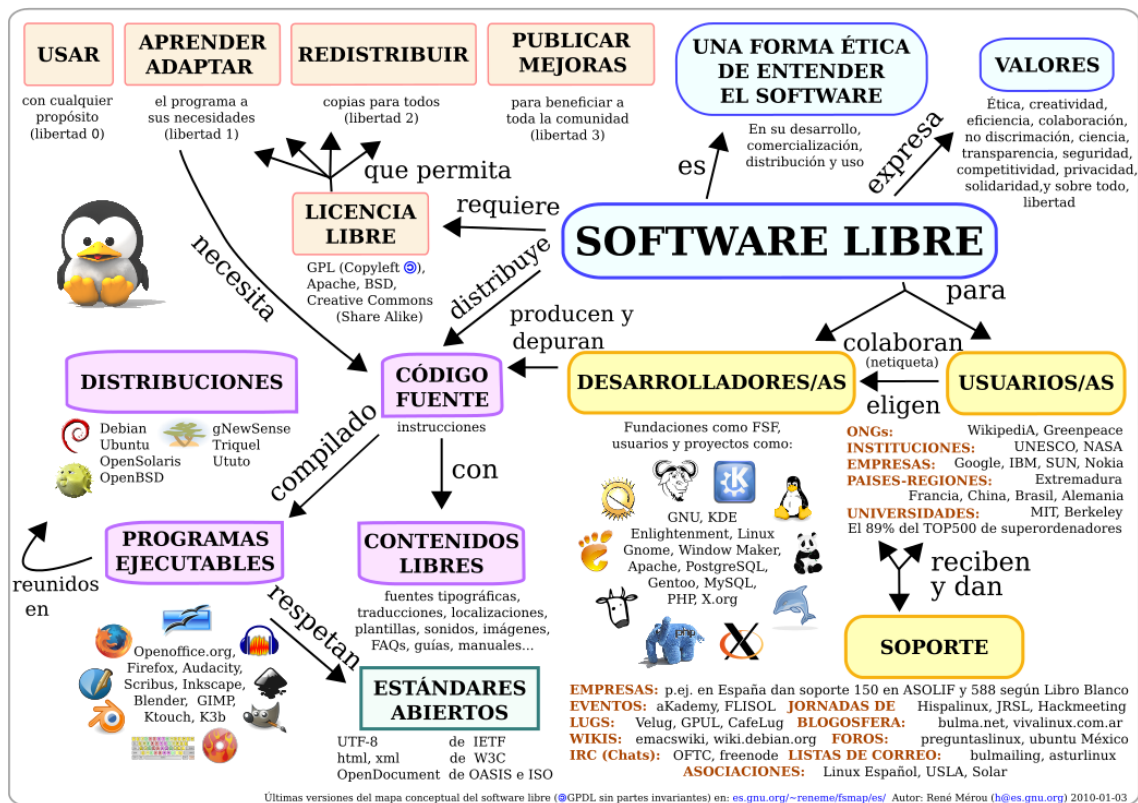


Figura 1: Mapa conceptual del Software Libre (René Mérou)

<http://www.vialibre.org.ar/>

Traducciones y localizaciones

Uso correcto de los dineros públicos

Transparencia de uso de la información pública

Gobierno electrónico

Hacktivism

Uso de computadoras y redes para promover fines políticos, principalmente libertad de expresión, derechos humanos y ética de la información

Empoderamiento de las minorías

Wikileaks

Diaspora

Proyectos Libres

Proyectos que generan conocimiento libre

Cualquier proyecto, sobre cualquier temática, ligado a licencias que permitan el uso, copia, modificación y distribución libre de los conocimientos o la información que allí confluyen. Unen a personas con iguales objetivos o problemáticas, que comparten trabajo y hacen públicos y libres sus resultados.

Libertades

- de usar el conocimiento generado en el proyecto para cualquier fin
- de estudiar el proyecto y adaptarlo a las propias necesidades usando la información generada por el proyecto
- de redistribuir copias de esa información, de manera que otros se beneficien de ella

- de mejorar el proyecto y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie de ellas
 - Open Source Hardware
 - Redes Libres
 - Project Gutenberg
 - Wikipedia, Wikimedia
 - Libros libres, música libre

Preguntas

1. ¿Cómo se autodefine la Free Software Foundation (FSF)?
2. ¿En qué consisten las cuatro libertades definidas por FSF?
3. ¿Puede imaginar una definición diferente de libertades? ¿Puede enunciar otro conjunto de libertades que garanticen los derechos del usuario considerados importantes por FSF?
4. ¿Cómo se resumen las diferencias entre código fuente y código binario, objeto o ejecutable? ¿Todo código ejecutable es no fuente? ¿Todo archivo fuente es de código?
5. ¿Por qué es importante la libertad de acceso al código fuente?
6. ¿Qué significa *privativo* en la terminología de FSF?
7. ¿De qué manera el software propietario o privativo vulnera las libertades establecidas por FSF?
8. ¿De qué manera las licencias libres impiden que las empresas que producen software propietario puedan apropiarse del trabajo de quienes desarrollan proyectos libres?
9. ¿Qué es Copyright? ¿En qué momento aparece el Copyright y a quién pertenece?
10. ¿Qué es Copyleft? ¿Cuál es la diferencia entre Copyright y Copyleft?
11. ¿Qué es *dominio público*?
12. El software Windows 8 es descargable de un sitio web¹. Verdadero o falso:
 - a) Luego, es software libre.
 - b) Debe pagarse para poder usarlo, luego no es software libre.
13. ¿Son ejemplos de Software Libre los siguientes?
 - Internet Explorer, Firefox, Chrome, Opera
 - Microsoft Office, Libre Office
 - Adobe Reader, evince, atril
 - Un ERP como Adempiere o SAP
 - Tango Gestión, de Axoft
 - MySQL, Microsoft SQL Server
 - El web server Apache, el web server IIS
 - Moodle, Joomla, Plone, Drupal
 - Android, OSX, ClearOS, Solaris
 - Aplicaciones que conozca para smartphones
 - Un juego web cualquiera que conozca
 - Extensiones de Chrome para aprender idiomas u otros utilitarios
 - Drivers para impresoras
 - El software web de administración de un router

¹<http://windows.microsoft.com/en-au/windows/download-shop>

2. Aspectos técnicos

Código fuente y ejecutables

- Lenguajes de programación
 - Lenguajes compilados
 - Lenguajes interpretados
- Formatos de archivos y documentos
 - Archivos legibles por humanos
 - Documentación: READMEs, Markup Languages, .pod, TeX/LaTeX, .rst...
 - Configuración o datos: XML, YAML, .cfg,...
 - Archivos estructurados
 - Formatos abiertos y propietarios
 - Formato .DOC, formato .odt
- Protocolos e interfaces
 - Protocolos de Internet vs. protocolos propietarios
 - RFCs
- Bibliotecas
 - Linking o vinculación
 - Linking dinámico, Shared Objects (.so), DLLs

Ciclo de compilación

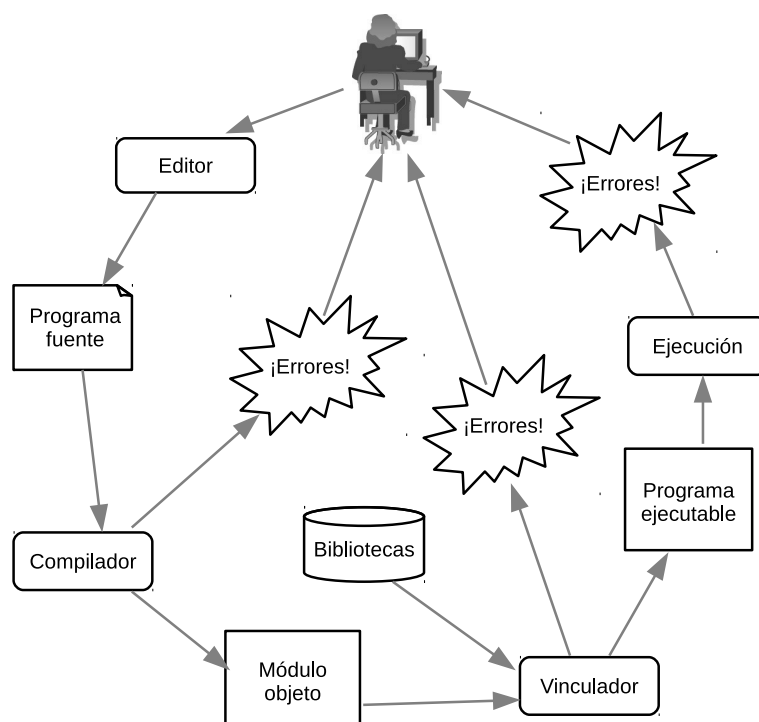


Figura 2: Ciclo de compilación y ejecución

```
#include <stdio.h>
int main() {
    printf("Hola, Mundo\n");
}
```

Figura 3: Código fuente en lenguaje C

[illegible]

Figura 4: Código objeto procedente del programa en C

```
#include <stdio.h>
int main() {
    hola();
    printf("Hola, Mundo\n");
    chau();
}
```

Figura 5: Un archivo fuente que usa dos funciones externas

```
#include <stdio.h>
int hola() {
    printf("Hola\n");
}

int chau() {
    printf("Chau\n");
}
```

Figura 6: Archivo fuente conteniendo funciones

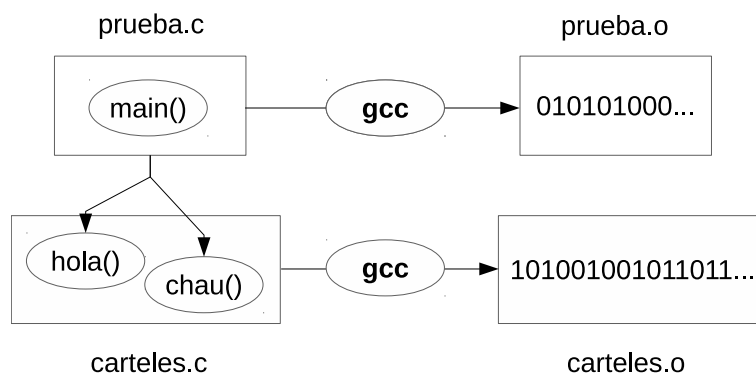


Figura 7: Compilación de los archivos fuente generando módulos objeto

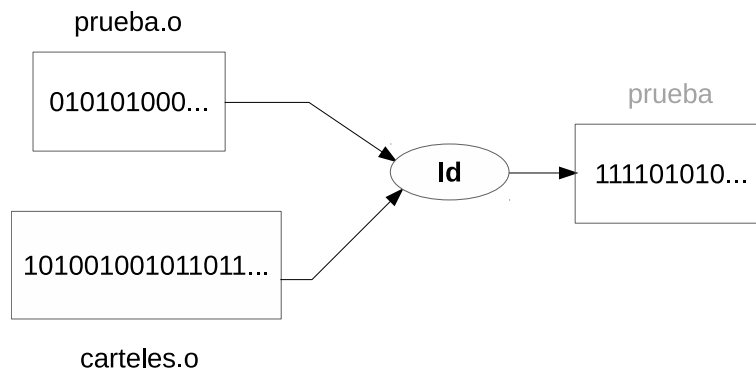


Figura 8: Vinculación estática de objetos generando un ejecutable

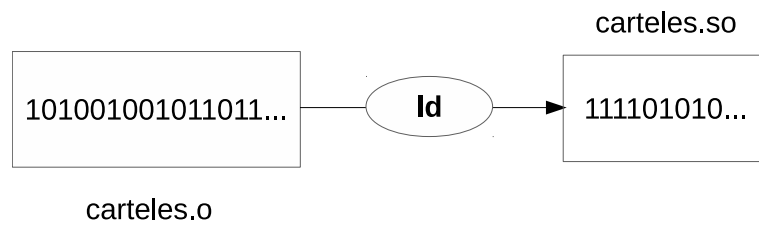
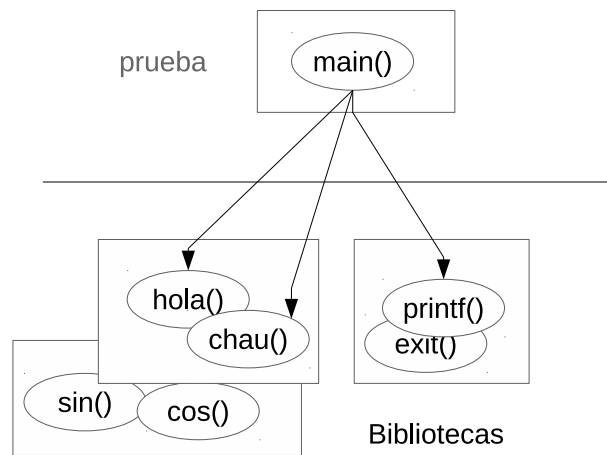
Figura 9: Creación de una biblioteca compartida (*shared object*)

Figura 10: Uso de bibliotecas de vinculación dinámica

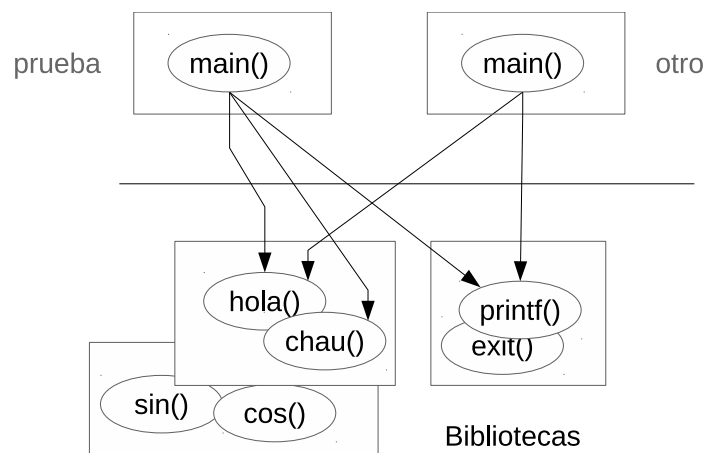


Figura 11: Compartiendo funciones de biblioteca

Proyectos de Software Libre

- Motivación
- Perfil de los desarrolladores
- Modelo de desarrollo
- Comunidad y vitalidad

La Catedral y el Bazar

- La Catedral y el Bazar vs. la Ingeniería de Software
- Representa la postura del movimiento Open Source
- Historia informal y razonada de un proyecto de SL
- Dos modelos de desarrollo
- Motivación y equipo de desarrollo inicial
- Reutilizar antes que desarrollar desde cero
- Roles de los usuarios
- Liberar rápido y con frecuencia

	Catedral	Bazar
Gobierno	Vertical	Democrático
Roles	Asignados Estáticos	Libres Móviles
Usuarios	Clientes	Co-desarrolladores
Procesos	Definidos	Multiestratégicos
Entregas	Planificadas Poco frecuentes	"Cuando esté listo" <i>Release early,</i> <i>Release often</i>

Infraestructura tecnológica

- Internet y RFCs
- Freecode <http://freecode.com>
- Sourceforge <http://sourceforge.net>
- Google Code <http://code.google.com>
- GitHub <http://github.com>

Preguntas

1. ¿Cómo se pueden resumir las diferencias entre *la catedral* y *el bazar*?
2. ¿Por qué en *La Catedral* y *El Bazar* se hace énfasis en la capacidad de reutilizar código? ¿Qué relación especial tiene esta capacidad con el SL?
3. ¿Cómo suele tener origen un proyecto de SL? ¿Cuál suele ser el disparador o motivador de la creación de un proyecto nuevo? ¿En qué casos se considera necesario comenzar un proyecto nuevo en lugar de aprovechar proyectos existentes?
4. ¿Qué roles asigna el modelo del Bazar a los usuarios del software?
5. ¿A qué se llama un *dictador benevolente*?
6. ¿Por qué el modelo del Bazar recomienda la regla "liberar rápido y con frecuencia"?
7. ¿En qué consiste la fase de captura de requerimientos de un proyecto de software? ¿En qué momento de la vida del proyecto se da esta fase?

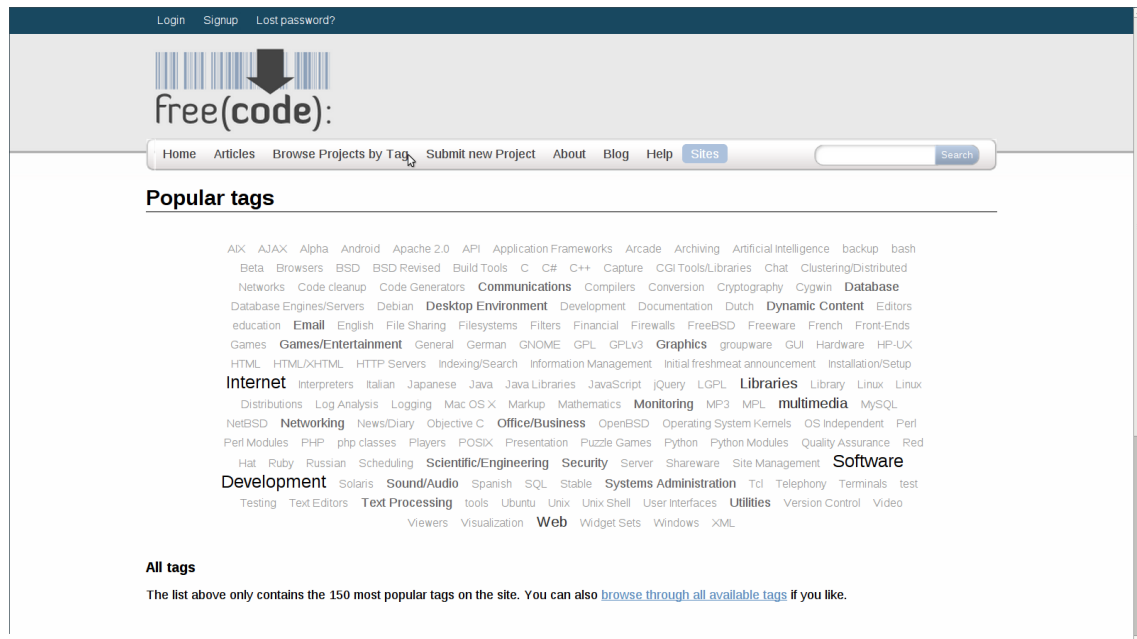


Figura 12: Nube de tópicos (tags) de Freecode.org

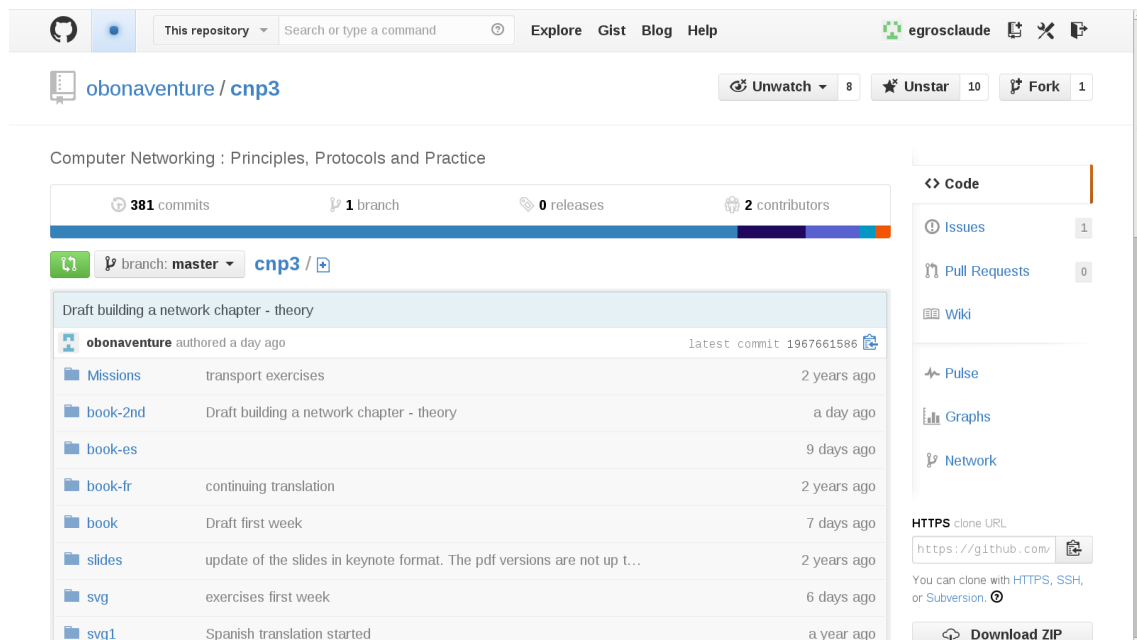


Figura 13: Interfaz del repositorio Github

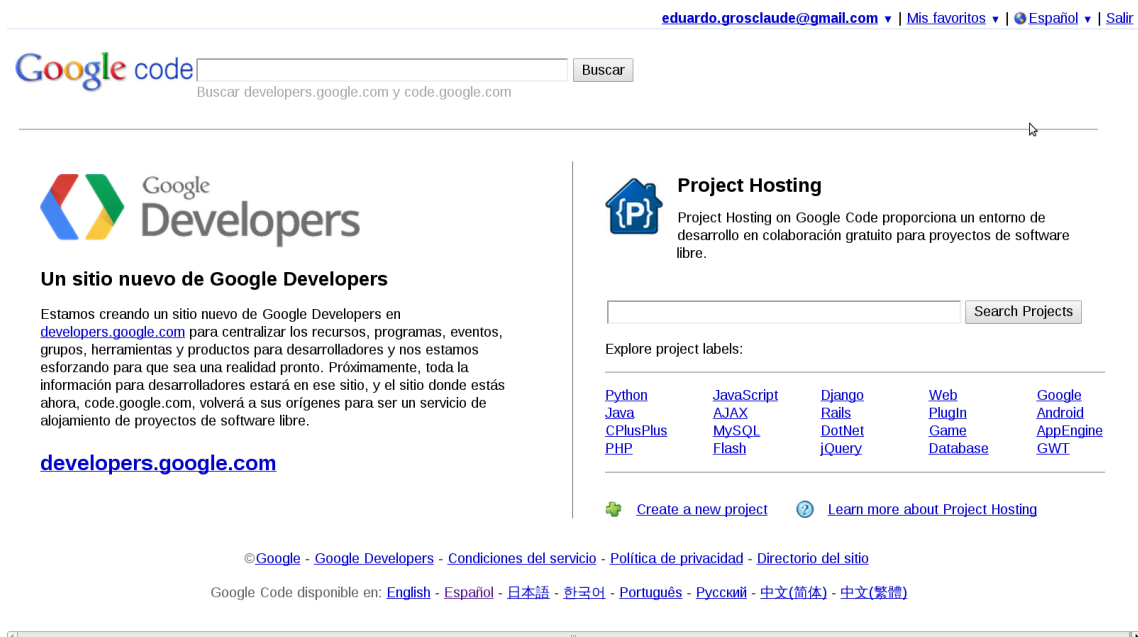


Figura 14: Interfaz del repositorio Google Code



Figura 15: Interfaz de edición del wiki de una distribución GNU/Linux

8. ¿A través de qué herramientas capturan requerimientos los proyectos de SL que usted conozca, en especial, los que su grupo está considerando para los trabajos prácticos de la materia?
9. ¿Qué herramientas de comunicación suelen utilizar los desarrolladores de SL? ¿Qué funciones cumplen estas herramientas (de comunicación interpersonal, de transmisión de conocimiento técnico, de transmisión o análisis de código, de descripción de procesos...)?
10. ¿En qué consiste la Ley de Brooks y por qué se dice que la complejidad de la interacción entre los desarrolladores se incrementa cuadráticamente?
11. Según Raymond, ¿de qué modo Internet y el SL se oponen a la Ley de Brooks? ¿A qué se llama el efecto Delphi?
12. ¿Qué crítica se hace a esta última afirmación de Raymond, según el material *Introducción al Software Libre* de Barahona y otros?
13. ¿Cuáles son los rasgos que hacen a los mejores líderes de proyecto, según Raymond?

3. Aspectos legales

Propiedad intelectual

- Copyright o derechos de autor
 - Comprenden la expresión de un contenido, no el contenido
 - Protección sobre la copia no autorizada
 - Aparece al momento de publicación
- Dominio público: no existe detentor de los derechos
- Derechos patrimoniales
 - Expiran en un cierto plazo luego de la muerte del autor
 - La obra pasa al dominio público
- Derechos morales
 - Son permanentes
- Propiedades protegidas
 - Obras literarias (software, caso especial), imágenes, filmaciones, composiciones musicales, otras

Licencias

- Cesión de derechos = licencia
- Contrato entre autor y usuarios
- Firma del contrato = instalar el producto
- Cesión de derechos de uso, nunca de propiedad
- Licencias restrictivas
 - Aseguran derechos del autor
 - Restringen derechos del usuario
- Licencias libres
 - Aseguran derechos del usuario
- Secreto comercial
 - Industria química, de alimentos
 - Ingeniería inversa, a veces prohibida
- Patentes
 - Una innovación es revelada públicamente
 - Haciendo posible su reproducción
 - A cambio se obtiene un monopolio temporario
 - Promueve la investigación privada, distribuyendo el producto de esa investigación y sin costos para el contribuyente
 - Pero puede dar lugar a una industria del juicio, parásita e improductiva
- Definición de innovación
 - Inventos
 - Algoritmos
 - Programas
 - Modelos de negocio
 - Sustancias naturales
 - Componentes de la vida
- Cultura Libre
 - http://es.wikipedia.org/wiki/Cultura_libre

Licencias de software

- Código, código fuente, código objeto, código ejecutable
- Documentación fuente y en otros formatos
- Concepto de Copyleft

<http://es.wikipedia.org/wiki/Copyleft>

Permitir la libre distribución de copias y versiones modificadas de una obra, exigiendo que en esas versiones se preserven los mismos derechos de uso

- Las licencias de FSF

http://es.wikipedia.org/wiki/GNU_General_Public_License

<http://www.gnu.org/licenses/gpl-faq.es.html>

- The Open Source Definition, de OSI

Licencias Protectivas y No Protectivas

http://es.wikipedia.org/wiki/Licencia_de_código_abierto

<http://opensource.org/docs/osd>

Licencias FSF

GPLv1

Permite la modificación y libre distribución de copias y requiere que se preserven las mismas libertades en las obras derivadas. La distribución del original o de las obras derivadas no debe impedir el acceso a los fuentes. Otras licencias bajo las cuales estén las obras no deben restringir cláusulas de la GPL.

LGPL

Con menos restricciones que GPL. Permite distribuir obras derivadas que se vinculan con software libre o propietario, como bibliotecas o drivers.

GPLv2

El cambio principal respecto de la GPLv1 consiste en que si el software tiene alguna restricción en conflicto con una parte protegida por GPL, no puede redistribuirse ("Libertad o Muerte").

GPLv3

("Anti-tivoización" y anti-acuerdos de patentes). Impide que un distribuidor asocie software GPL a un dispositivo que no permite la modificación de dicho software. Impide que una obra sea entregada a un distribuidor que cobre por el trabajo del autor. Formaliza los permisos adicionales que conceden los autores. Modifica la terminación de la licencia por violación (simplifica el "pedir perdón" a los propietarios).

- Tivoización

Creación de un sistema que usa software protegido por copyleft pero incluye hardware que restringe la libertad de los usuarios de correr obras derivadas de ese software. Es ilegal bajo la GPLv3.

Affero GPL o AGPL

GPLv2 más cláusula que obliga a distribuir los fuentes del software si éste se utiliza ofreciendo servicios a través de una red.

Licencias tipo BSD

- “Minimalistas”
- Originadas en proyectos de extensión de universidades, no garantizan la libertad del software
- Las redistribuciones en fuente o binarios deben preservar la nota de copyright
- No se puede usar el nombre del propietario para promocionar obras derivadas
- El propietario no se responsabiliza por el uso del programa

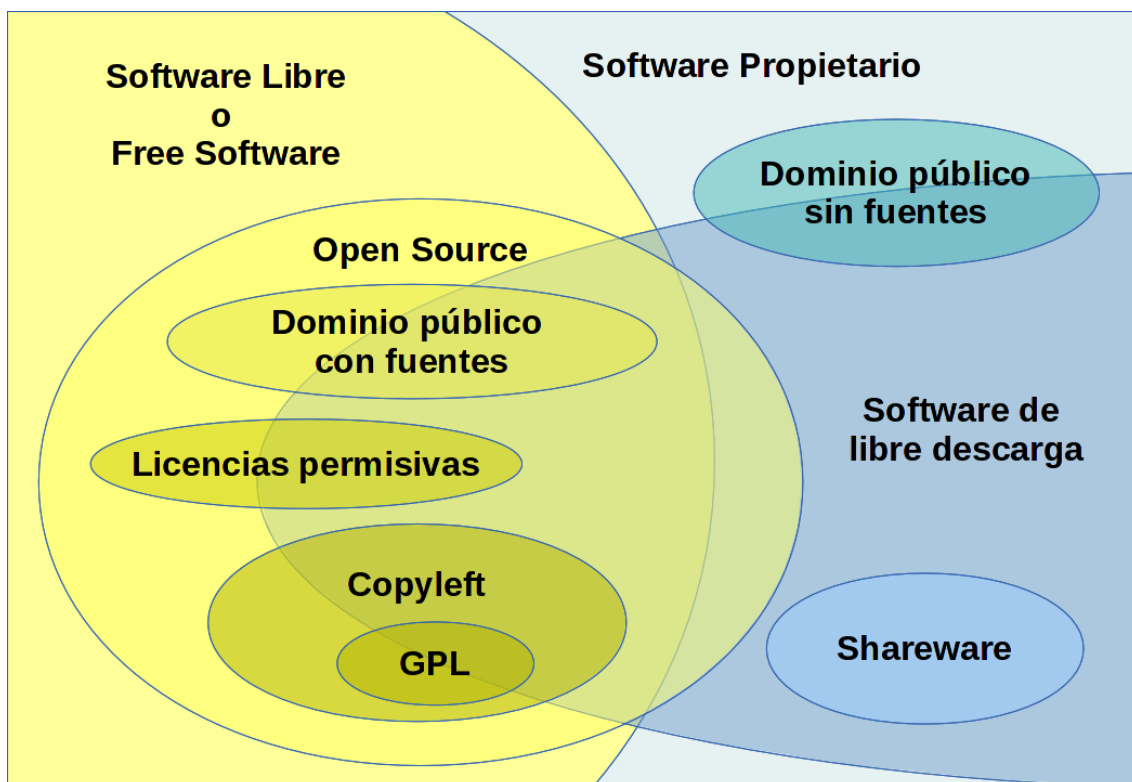








Figura 16: Categorías de software según FSF

Licencias de contenidos libres

- Creative Commons
- Dimensiones
 - Atribución
 - Uso comercial
 - Obras derivadas
 - Compartir igual
- <http://www.creativecommons.org.ar/>
- Elegir una licencia: <http://creativecommons.org/choose/?lang=es>

Licencia	Atribución	Uso comercial	Obras derivadas	Compartir igual	
CC-BY	●	●	●		
CC-BY-NC	●		●		
CC-BY-NC-ND	●				
CC-BY-NC-SA	●		●	●	
CC-BY-ND	●	●			
CC-BY-SA	●	●	●	●	

Preguntas

1. El acceso al código fuente, ¿implica automáticamente la posibilidad de introducir modificaciones y mejoras? ¿Y la posibilidad de redistribuir el software?
2. Si modifico un software con licencia libre, ¿estoy *obligado* por la licencia libre a redistribuirlo públicamente?
3. ¿Cómo se define el concepto de Copyleft?
4. Al licenciar una obra bajo alguna forma de Copyleft, ¿se pierde el Copyright?
5. En caso de redistribuir software liberado bajo licencia libre, ¿a qué me obliga una licencia libre? ¿A qué se obliga la persona que lo recibe de mí?
6. Si una empresa cobra por la distribución de software liberado bajo licencia libre, ¿está violando la licencia?
7. Si descargo un software publicado bajo una licencia libre, ¿puedo redistribuirlo? ¿Puedo cobrar por redistribuirlo? ¿Y si se trata de una licencia Open Source?
8. Si descargo un software publicado bajo una licencia libre, ¿puedo modificarlo? ¿Puedo cobrar por modificarlo? ¿Y si se trata de una licencia Open Source?
9. ¿Cuál es la motivación de FSF para asumir el copyright de las obras libres?
10. ¿En qué consisten las licencias de software “minimalistas” o permisivas y qué ejemplos se pueden dar?
11. ¿Cuál es la opinión de FSF respecto del movimiento Open Source?
12. ¿Cómo puede resumirse la diferencia entre los conceptos de Free Software y Open Source?
13. Mencione proyectos o productos Open Source que no sean considerados libres por FSF.
14. ¿Cuáles son las principales licencias que reconoce FSF como libres?
15. ¿A qué se llama el “efecto viral” de la GPL?
16. ¿Cuál ha sido la historia de versiones de GPL hasta el momento y qué motivó cada versión? ¿Considera que puede ser necesaria una nueva versión en algún momento futuro?
17. ¿Por qué se creó la licencia LGPL? ¿En qué casos se recomienda usar o no usar la LGPL?
18. ¿Por qué no es lo mismo decir Software No Libre que Software Comercial? ¿Por qué no es lo mismo Licencia Libre que Dominio Público?
19. ¿Qué es MP3? ¿Qué clase de protección legal tiene MP3 y qué consecuencias tiene para los usuarios? ¿Existen alternativas técnicamente equivalentes y legalmente menos restrictivas?
20. ¿A qué se llama en inglés *appliances*? ¿Qué ejemplos conoce?
21. ¿Qué es el producto Tivo y de qué manera se relaciona con la historia de la licencia GPL?
22. ¿Cuándo un dispositivo *appliance* respeta las libertades, según FSF?

23. ¿Puede identificar desarrollos originalmente libres que fueron absorbidos por intereses corporativos generando software no libre? ¿Qué licencias utilizaron?
24. ¿Cuáles son las características de las licencias tipo BSD? ¿Qué proyectos de SL conocidos las emplean?
25. ¿Puede mencionar dispositivos *appliance* que respeten y no respeten las libertades según FSF?
26. Si una empresa crea una modificación de un producto de SL y lo utiliza en forma interna, ¿debe publicar los fuentes de la obra derivada? ¿Y si ofrece servicios a través de la red utilizando esa obra derivada?
27. ¿Qué significa *jailbreak*? ¿Qué es “rooting”? ¿Qué significa “ingeniería inversa”? ¿Es una actividad legal?
28. ¿Qué significa DRM? ¿De qué manera afecta DRM a las libertades del usuario según FSF? ²
29. ¿Qué dimensiones comprenden las licencias Creative Commons? ¿Cuál de las dimensiones corresponde al concepto de Copyleft según FSF? ¿Qué licencias CC no incluyen Copyleft?
30. ¿Qué es WIPO? ¿Qué opinión sobre Creative Commons volcó la Argentina en su propuesta, conjunta con Brasil, a la WIPO en 2004? ¿A qué se refiere esta propuesta con “medidas tecnológicas de protección en el entorno digital”? ¿A qué se denomina “la dimensión del desarrollo”? ³
31. ¿Existen las patentes de SW en Argentina? ⁴. ¿Cuáles son los peligros de un sistema de patentes “permisivo”?

²http://es.wikipedia.org/wiki/Gestión_digital_de_derechos

³http://www.wipo.int/edocs/mdocs/govbody/es/wo_ga_31/wo_ga_31_11.pdf

⁴<http://www.vialibre.org.ar/mabi/4-software-patentado.htm>

4. Uso y aplicación de Software Libre

Selección de Software Libre

- Cualidades del software
 - Proyecto vivo y en actividad
 - Proyecto más aceptado
 - Estable y maduro
 - Tendencia
- Cualidades de la organización
 - Impacto que provocará el cambio
 - Estudio de Costos/Beneficios
 - $C_1/B_1 \longleftrightarrow C_2/B_2$
 - Estudio de Costo Total de Propiedad de ambas soluciones
 - Riesgos
 - Taxonomía de riesgos (ver Anexo C)
- Valuación de costos, beneficios, riesgos
 - Difícil en forma absoluta
 - Cuando se trata de comparar se puede fijar una escala adimensional
 - Los riesgos se pueden computar como *probabilidad de ocurrencia * gravedad*

Administración de TI

- Frameworks de administración de Tecnologías de Información (TI)
- Compendios de mejores prácticas
 - CobiT
 - TI en general
 - Libremente reproducible
 - ITIL
 - Servicios
 - Más detallada
 - Licencia restrictiva
 - ISO

CobiT

- CobiT define 34 procesos de TI. Incluye herramientas para medición de desempeño, una taxonomía de factores críticos de éxito, y modelos de madurez para asistir en la toma de decisiones hacia el mejoramiento de capacidades.
- Objetivos de CobiT
 - Asegurar que
 - TI se alinea con la actividad de la organización
 - TI habilita las operaciones y maximiza los beneficios
 - Los recursos se usen responsablemente
 - Los riesgos se manejen adecuadamente
- Dominios definidos en CobiT
 - Plan and Organise (PO)
 - Acquire and Implement (AI)
 - Deliver and Support (DS)

Monitor and Evaluate (ME)
“Acquire and Implement, AI”
Identify Automated Solutions
Acquire and Maintain Application Software
Acquire and Maintain Technology Infrastructure
Enable Operation and Use
Procure IT Resources
Manage Changes
Install and Accredite Solutions and Changes

Acquire and Implement AI2

Acquire and Maintain Application Software

MANAGEMENT GUIDELINES

AI2 Acquire and Maintain Application Software

From	Inputs	Outputs	To
P02	Data dictionary; data classification scheme; optimised business system plan	Application security controls specification	DS5
P03	Regular 'state of technology' updates	Application and package software knowledge	AI4
P05	Cost-benefits reports	Procurement decisions	AI5
P08	Acquisition and development standards	Initial planned SLAs	DS1
P010	Project management guidelines; detailed project plans	Availability, continuity and recovery specification	DS3 DS4
AI1	Business requirements feasibility study		
AI6	Change process description		

RACI Chart

Functions

Activities

	CEO	CFO	Business Executive	CIO	Business Process Owner	Head Operations	Chief Architect	Head Development	Head IT Administration	PMO	Compliance, Audit, Risk and Security
Translate business requirements into high-level design specifications.				C	C	A/R		R	C		
Prepare detailed design and technical software application requirements.			I	C	C	C	A/R		R	C	
Specify application controls within the design.				R	C		A/R		R	R	
Customise and implement acquired automated functionality.				C	C		A/R		R	C	
Develop formalised methodologies and processes to manage the application development process.			C		C	C	A	C	R	C	
Create a software QA plan for the project.				I	C	R		A/R	C		
Track and manage application requirements.						R		A/R			
Develop a plan for the maintenance of software applications.			C		C	A/R		C			

A RACI chart identifies who is Responsible, Accountable, Consulted and/or Informed.

Figura 17: Un ejemplo de guía de actividades de CobIT

Migraciones en general

- Difícil por la naturaleza de los sistemas privativos
- Apoyo de los usuarios
 - Se necesitan defensores del cambio
 - Información para involucrarlos
- Diagnosticar claramente el escenario de partida
 - Hardware, software, configuración de la red, usuarios
- Definir la situación final deseada

- Justificar la migración
 - Análisis de costos de la migración y proyección de ahorros factibles
- Planificar la migración
- Migración de los datos
- Preparar y capacitar el equipo técnico

Formas de migración a SL

- Repentina
 - Riesgosa
 - Requiere gran inversión en capacitación
- Progresiva
 1. Usando SL bajo el sistema operativo existente
 2. Migrando los datos progresivamente
 3. Migrando los ambientes y sistemas operativos

Facilitar migración futura

1. Usar Formatos y estándares abiertos
2. Usar metodologías basadas en capas para el desarrollo, separando código de interfaz y de acceso a los datos
3. Generar aplicaciones portables, evitando lenguajes de arquitecturas específicas
4. Evitar la construcción de aplicaciones que requieran otras privativas
5. Desarrollos web que cumplan estándares W3C y validados contra navegadores libres

Costo Total de Propiedad, CTP

- O también *Total Cost of Ownership, TCO*
- Cálculo para facilitar la evaluación de costos directos e indirectos asociados con la adopción de un componente de IT
- Evaluar beneficios de la migración frente a los costos
- El CTP define el costo total de la propiedad para el uso de una tecnología concreta durante el período de vida de dicha tecnología
 - Medición del impacto
- Taxonomías de costos
 - Costos de adquisición
 - Costos asociados con reparaciones
 - Costos de oportunidad
 - Costos extendidos
 - Servicios conexos de soporte, redes, seguridad, capacitación, licenciamiento de software u otros componentes
- ¿Taxonomías de *beneficios*?

Metodologías de análisis de CTP

■ Una metodología genérica

<http://blogs.msdn.com/b/eduardop/archive/2006/05/29/610441.aspx>

Considera costos divididos en iniciales y operativos

Costo inicial de la solución $CI = CH + CS + CINS + CCON$

CH = Costo del hardware

CS = Costo del software

CINS = Costo de servicios iniciales de instalación

CCON = Costo de servicios iniciales de configuración

Costo de administración (CA)

Personal dedicado al mantenimiento operativo

Cálculos basados en honorarios y gastos por personal

Costo de operación (CO)

Pérdidas por caída de operación o soporte reactivo

Cálculos basados en costo de incidentes

Costo de soporte (CS)

Costo de proveer asistencia a usuarios

Basado en costo por hora de soporte

TV = Tiempo de vida de la solución

Costo total = $CI + (CA + CO + CS) * TV$

■ Otra metodología orientada a SL

Centro de Excelencia en Software Libre U. Castilla-La Mancha

http://www.bilib.es/documentos/Taller_de_Migracion.pdf

Costo Total CT = CD + CI = Costos Directos + Costos Indirectos

Directos se producen como consecuencia de la adquisición del software

Indirectos se producen como consecuencia de pérdidas o caídas de productividad

$CD = CH + CS + CSOP + CFOR + CPER$

CH = Costo del hardware

CS = Costo de licenciamiento de software

CSOP = Costo de soporte = Instalación + Configuración + Mantenimiento

CFOR = Costo de formación del personal de operación

CPER = Costo de personal de operación

$CI = CM + CC + CSEG + CE + CDISP$

CM = Costo de mantenimiento por errores o problemas del software

CC = Costo de oportunidad debido a caídas de sistema

CSEG = Costo de aseguramiento informático

CE = Costo de asegurar la escalabilidad

CDISP = Costo de asegurar la disponibilidad

5. Producción de y con Software Libre

Adopción estratégica de Software Libre

1. Usar SL
2. Reportar bugs o solicitudes de características
3. Aportar patches o código
4. Convertirse en miembro de la comunidad de usuarios y desarrolladores

Desarrollo de proyectos de Software

- Metodologías de análisis y diseño

- Etapas básicas

- Estudio de viabilidad

- En esta fase se considera si el proyecto se puede realizar, teniendo en cuenta las circunstancias internas y externas, las diferentes soluciones posibles y los recursos de los cuales se dispone.

- Análisis

- Se analizan las necesidades que se desea satisfacer con el nuevo proyecto, se ajustan los objetivos finales y se centra la solución tecnológica. También en esta fase se definen las interfaces entre los diferentes subsistemas que formarán el proyecto y las de usuario que permitirán interactuar con el sistema.

- Diseño

- En esta fase se realiza el diseño tecnológico de la solución escogida, proponiendo una arquitectura global y analizando y estudiando todos los casos de usos (o los más representativos) existentes.

- Desarrollo

- En esta fase se construye la solución propuesta teniendo en cuenta el entorno utilizado, se escogen las licencias, se genera la documentación y se ejecutan las pruebas acordes al tipo de proyecto y metodología utilizada.

- Implantación

- Se traspa del entorno de desarrollo al sistema real y se realizan todas las pruebas y medidas de niveles de prestaciones que conducirán a la aceptación definitiva de proyecto. Se define también el plan de mantenimiento y se toman las decisiones adecuadas para el correcto funcionamiento del sistema durante el resto de su vida.

Proyecto de software tradicional vs. SL

- Modos de desarrollo

- Interno

- Binarios -ocasionalmente fuentes- liberados una vez que se ha validado internamente el desarrollo
 - Soporte via foros propios o sistema web de tickets
 - Modificaciones disponibles luego de testing interno, altas demoras
 - Bases de datos de bugs disponibles solamente para la organización
 - Desarrollo, validación, soporte, llevados a cabo por el proveedor

- Abierto

- Administración de fuentes a través de repositorios públicos (CVS, SVN, GIT)
 - Soporte via foros, listas, IRC, etc.
 - Modificaciones disponibles en las listas y repositorios

- Páginas de proyecto como documentación básica para los usuarios
- Implica relacionarse con nuevas tecnologías y modelos de desarrollo
- Modos de valuación
 - Precio del software tradicional en función de un costo en horas de desarrollo
 - El desarrollo del SL es soportado por la comunidad
 - Luego desaparece la preocupación por el costo de desarrollo
 - Desplazar el interés económico de la programación a los servicios
 - Caso extremo: productos ERP

Modelo de desarrollo Open Source

- El desarrollo está distribuido entre múltiples equipos
- Trabajan en diferentes ubicaciones
- Estructura flexible, resistente a llegadas o partidas de desarrolladores
- Procesos definidos para incorporar e integrar código
- Comunicación autodocumentada
- Desarrollo asincrónico
- Características del producto son desarrolladas incrementalmente

Modelos de negocio

- Roles alternativos
 - Desarrollo de SL
 - Capacitación, instalación, soporte, personalización, consultoría, apoyo
 - Integración de sistemas
 - Distribución de SL (RedHat, Novell)
 - Distribución de accesorios (libros, CDs)
- Monetización del desarrollo o servicios de distribución
 - Publicidad on line
 - Partnership con empresas de Internet
 - *Crowdfunding, perking*
- <http://www.bilib.es/recursos/documentacion/>

Motivaciones para producir SL

- Partir de una solución parcial preexistente
 - Facilitar la puesta en marcha del proyecto
 - Ahorro en costos de análisis y desarrollo
 - Aprovechar la plataforma de usuarios y desarrolladores
 - Acceder a testing masivo y gratuito
- Influir en la agenda de desarrollo del proyecto
 - Upstreaming
 - Roles de mantenedor, traductor, documentador, etc.
 - Darle persistencia o diferentes alcances al proyecto
- Compartir el esfuerzo de desarrollo con pares
 - Empresas del mismo ramo, organizaciones del mismo nivel
 - Capitalizar las ideas, descubrimientos y desarrollos de los demás

SL en las organizaciones

Administración pública

- La AP administra registros de información que son propiedad de los ciudadanos
- Tiene la responsabilidad de custodiarlos y garantizar su privacidad, integridad, persistencia y accesibilidad
- Debe seleccionar las mejores herramientas tecnológicas para implementar la política de información, incluyendo aplicaciones, protocolos y formatos de datos
- Debe ser cuidadoso en sus inversiones en herramientas tecnológicas
- Debe aprovechar los nichos de funcionalidad común en las administraciones locales o regionales, compatibilizando y compartiendo una base de software e información
- *La Administración Pública como receptora de proyectos internos de software libre*, Francesc Rambla i Marigot, 2009, UOC ⁵
- *Razones por las que el Estado debe usar Software Libre*, Federico Heinz, 2001 ⁶

Sector privado

- Objetivos
 - Competir con líderes de mercado en posición ventajosa
 - Aumentar la participación en un mercado
 - Investigar la aceptación del mercado hacia un producto, reduciendo riesgos
 - Aprovechar promoción gratuita generada por la libre distribución
 - Desarrollar un sistema de soporte a bajo costo
 - Acceder a desarrolladores capacitados
 - Reducir el tiempo de llegada a los mercados
- Modelos de Negocio
 - Doble Licenciamiento

Este modelo se basa en la distribución de un producto bajo dos licencias distintas: una licencia propietaria tradicional, y una licencia libre restrictiva (tipo GPL). De esta manera, si alguien quiere generar un trabajo derivado, y redistribuirlo sin el código, puede hacerlo, pero deberá pagar una licencia. De lo contrario, todos los trabajos derivados deben redistribuirse con el código.
 - *Open Core* o núcleo abierto

En este modelo existen dos versiones distintas de un programa, una versión básica libre, y una versión comercial propietaria, basada en la anterior, pero con funcionalidad adicional implementada a través de plugins o accesorios. La versión libre debe emplear una licencia de tipo MPL o BSD, que permita la combinación para crear un producto cerrado.
 - Software como Servicio (SaS)

Las empresas desarrolladoras de un producto también podrán explotarlo mediante el paradigma de software como servicio. En lugar de ofrecer servicios de instalación y soporte, la empresa se hace cargo de toda la infraestructura de hardware y software, ofreciendo directamente la funcionalidad a través de Internet. Los ingresos generados, de naturaleza recurrente, toman la forma de suscripciones al servicio.
 - Capacitación y servicios conexos

Capitalizar la experiencia de los líderes de proyecto ofreciendo servicios conexos a los productos libres.

⁵<http://cv.uoc.edu/autors/MostraPDFMaterialAction.do?id=154680>

⁶<http://proposicion.org.ar/doc/razones.html>

Parte III

Anexos

A. Licencias libres y no libres

Comparación de GPL y MS EULA

EULA: End User License Agreement

GPL: General Public License

<http://blog.desdelinux.net/eula-windows-vs-gpl-linux-duelo-de-licencias>

Características básicas de la EULA

Prohibida su copia y redistribución (copyright). Puede ser usada por una sola computadora con un máximo de dos procesadores. No puede ser utilizado como servidor web o como servidor de archivos. Requiere registro después de 30 días. Podría dejar de funcionar si se realizan cambios de hardware. Las actualizaciones pueden cambiar la EULA si la compañía así lo decidiera. Puede ser transferida al nuevo usuario una sola vez. El nuevo usuario debe estar de acuerdo con los términos de uso (EULA). Impone limitaciones a la reingeniería inversa. Se conceden permisos a Microsoft para tomar información sobre el Sistema y su uso. Se conceden permisos a Microsoft para proveer esta información a otras organizaciones. Se conceden permisos a Microsoft a realizar cambios a el sistema sin el consentimiento del usuario. Garantía por los primeros 90 días, Actualizaciones, reparaciones y parches no tienen garantía.

Características básicas de la licencia libre GPL

Libertad de copiar, modificar y redistribuir el software. Impide que un grupo o ente impida que otro grupo o ente no pueda tener estas mismas libertades. Provee cobertura a los derechos de los usuarios de copiar, modificar y redistribuir el software. Se puede vender si el usuario así lo decide, y los servicios conexos a dicho software pueden ser cobrados. Toda patente debe ser licenciada para el uso de todos o no ser licenciada en absoluto. Software modificado no debe llevar costo de licencias. Se debe proveer con el código fuente. Si hay un cambio en la licencia, los términos generales de la licencia existente se mantienen.

B. Política de uso de SL en la UNC

Establecen la política de uso de Software Libre en la UNCo

Martes, 13 de Diciembre de 2011 12:34

Última actualización el Martes, 13 de Diciembre de 2011 20:10

Escrito por Prensa UNCo

Mediante la Ordenanza N° 590 del 13 de diciembre de 2011, el Consejo Superior de la Universidad Nacional del Comahue resolvió establecer como política en el ámbito administrativo de la Casa de Altos Estudios el uso de Software Libre desarrollado con estándares abiertos en todos sus sistemas y equipamientos informáticos.

La medida fue tomada luego de que la Facultad de Informática elevara una solicitud en ese sentido.

En el Artículo 2 de la Ordenanza N° 590/2011, se brindan cuatro conceptos claves para la normativa.

El primero de ellos es Software Libre que es entendido como **Programa de computación cuya licencia garantiza al usuario acceso al código fuente del programa y lo autoriza a ejecutarlo con cualquier propósito, modificarlo y redistribuir tanto el programa original como sus modificaciones en las mismas condiciones de licenciamiento acordadas al programa original, sin tener que pagar regalías a los desarrolladores previos.**

El segundo es Software de Código Abierto, definido como **Programa de computación cuya licencia garantiza al usuario acceso al código fuente del programa y lo autoriza a ejecutarlo con cualquier propósito, modificarlo y redistribuir tanto el programa original como sus modificaciones en las mismas condiciones de licenciamiento acordadas al programa original, sin imponer restricciones en otro software que distribuya junto con el mismo.**

El tercero es Software Privativo, entendido como **Programa de computación cuya licencia establece restricciones de uso, redistribución o modificación por parte de los usuarios, o requiere de autorización expresa del Licenciador.**

El cuarto es Estándares Abiertos, definidos como **Especificaciones técnicas, publicadas y controladas por alguna organización que se encarga de su desarrollo, las cuales han sido aceptadas por la industria, estando a disposición de cualquier usuario para ser implementadas en un software libre u otro, promoviendo la competitividad, interoperatividad o flexibilidad.**

Por último, en el Artículo 3 se recomienda el empleo prioritario del software que garantice las libertades de ejecución, modificación y distribución con estándares abiertos en todas las actividades de la Universidad del Comahue, adoptando siempre que haya una alternativa el siguiente orden de prioridad:

a) Software Libre. b) Software de Código Abierto. c) Software privativo que respete los estándares abiertos.

Considerandos

En la Ordenanza se destaca que la Universidad Nacional del Comahue es depositaria de datos e información generada por la propia administración, miembros de la comunidad universitaria, otras instituciones universitarias, instituciones gubernamentales, organizaciones del tercer sector, empresas y ciudadanía en general.

Asimismo se considera que es responsabilidad y obligación de la gestión de gobierno controlar la seguridad, confiabilidad e interoperabilidad de la información que recibe, procesa y remite.

El empleo de formatos cerrados genera una dependencia tecnológica interminable hacia el proveedor de turno haciendo que el propio generador de la información requiera subordinarse a una aplicación sobre la que no tiene control para acceder a sus propios datos, por lo cual es necesario que se termine con la misma, implementando sistemas que permitan mantenerse en el mundo informático sin necesidad de depender de un proveedor.

En este sentido, el Software Libre permite la implementación de sistemas operativos, formatos y aplicaciones que podrán ser libremente utilizados y modificados cuando las necesidades lo requieran.

Actualmente, existen programas pueden reemplazar sus respectivos pares de software privativo para todas las actividades administrativas que realiza la Institución educativa.

De hecho, en la UNCo hay capacidad técnica para llevar esto a cabo.

Cabe destacar que ya se han generado iniciativas en algunas Unidades Académicas y dependencias de esta universidad y de otras universidades nacionales y que el Grupo de Trabajo sobre Tecnologías de la Información y la Comunicación de la UNCo, conformado por la Secretaría General, la DTI, la UAI y la Facultad de Informática avala la propuesta.

Además, desde el Rectorado se estableció e implementó un programa de capacitación y migración al Software libre.

Por último, se da cuenta en la Ordenanza que es necesario establecer un marco jurídico para fijar políticas en el área informática.

C. Cuestionario para identificación de riesgos operativos

Tomado de **A Taxonomy of Operational Risks**, CMU/SEI-2005-TN-036 - <http://www.sei.cmu.edu/reports/05tn036.pdf>

Taxonomía de Riesgos

A - Misión

Riesgos a la operación que pueden surgir a causa de la naturaleza de la misión que se intenta cumplir.

1. Tareas, órdenes, planes

¿Hay riesgos que puedan surgir de la forma en que se distribuyen las tareas, en que se dan las órdenes, o en la que se desarrollan los planes operativos? (Estabilidad, completitud, claridad, validez, factibilidad, precedencia, oportunidad)

2. Ejecución de la misión

¿Hay riesgos que puedan surgir de la ejecución de la misión? (Eficiencia, efectividad, complejidad, oportunidad, seguridad)

3. Producto o servicio

¿Hay riesgos que puedan surgir del producto final o servicio que provee esta misión operativa? (Usabilidad, efectividad, oportunidad, exactitud, correctitud)

4. Sistemas operativos

¿Hay riesgos que puedan surgir de los sistemas operativos usados? (Productividad, adecuación, usabilidad, familiaridad, confiabilidad, seguridad, inventario, instalaciones, soporte de sistemas)

¿Hay otros riesgos que puedan surgir de la misión, que no estén cubiertos por las categorías anteriores?

B - Procesos

Riesgos a la misión que puedan surgir de la forma en que la organización ejecuta la misión.

1. Procesos operativos

¿Hay riesgos que puedan surgir del proceso que ha elegido la organización operativa para ejecutar la misión? (Formalidad, adecuación, control de procesos, familiaridad, control de producto)

2. Procesos de mantenimiento

¿Hay riesgos que puedan surgir del proceso que utiliza la organización de mantenimiento para mantener los sistemas operativos? (Formalidad, adecuación, control de procesos, familiaridad, calidad de servicio)

3. Procesos de administración

¿Hay riesgos que puedan surgir de la forma en que se planifica, monitoriza o controla el presupuesto operativo, o en la estructura operativa de la organización, o en su forma de manejar interfaces internas y externas? (Planeamiento, organización, experiencia en administración, interfaces de los programas)

4. Métodos de administración

¿Hay riesgos que puedan surgir de la forma en que se maneja el personal operativo? (Monitorización, administración de personal, aseguramiento de calidad, administración de configuración)

5. Ambiente de trabajo

¿Hay riesgos que puedan surgir del ambiente general, o de la organización a la cual pertenece la unidad operativa? (Actitud hacia la calidad, cooperación, comunicación, moral)

¿Hay otros riesgos que puedan surgir de la forma en que la unidad operativa enfrenta la misión, pero que no estén cubiertos por las categorías mencionadas?

C - Restricciones

Riesgos a la misión que puedan surgir de orígenes fuera de nuestro control.

1. Recursos

¿Hay riesgos que puedan surgir de recursos que necesite la organización operativa pero que esté fuera de su control obtener o mantener? (Agenda, personal, presupuesto, instalaciones, herramientas)

2. Políticas

¿Hay riesgos que puedan surgir de políticas legalmente vinculantes o restrictivas? (Leyes y reglamentos, restricciones, obligaciones contractuales)

3. Interfaces

¿Hay riesgos que puedan surgir de interfaces externas que la organización operativa no pueda controlar? (Comunidad de usuarios o clientes, agencias asociadas, contratistas, líderes mayores, vendedores, políticas)

¿Hay otros riesgos que puedan surgir de factores fuera del control de la organización operativa, pero que no estén cubiertos por las categorías mencionadas?