

# Impacto del Software Libre en las empresas

Ricardo Galli

Dept. Matemàtiques i Informàtica  
Universitat de les Illes Balears

BULMA  
Grup d'Usuaris de Linux de Balears

# Estructura de la presentación

- Software libre, software propietario, historia #3.
- Las TI en las empresas #12.
- Sobre la motivación de los programadores y los consumidores #26.
- El SL en las empresas #32.
- Desarrollo de SL en las empresas #38.
- El SL y la “innovación” #45.
- El negocio informático #50.
- Conclusiones #54.

# Introducción

Definición de software libre, el software propietario,  
historia

# Definición de Software Libre

- Libertad de ejecutar el programa para cualquier propósito
- Libertad para estudiar el programa y adaptarlo
- Libertad para distribuirlo por cualquier medio
- Libertad para mejorarlo y distribuir las modificaciones
  - Puede ser y tener usos comerciales
  - El *Copyleft* asegura que siga siendo libre

# Historia del software

- El software no fue un “producto de mercado” hasta hace 25 años
- Los desarrollos en *mainframes*, notoriamente en el MIT y Berkeley eran abiertos y compartidos entre programadores
  - Pero en 1969, y debido al juicio anti-monopolio, IBM empieza a comercializar hardware y software por separado
- En los 70 se empieza a desarrollar la microinformática: *Homebrew Computer Club*
- Unix

# El nacimiento del software propietario

- *¿Pero qué es ésto? Como muchos de los hobistas ya debéis saber, la mayoría de vosotros robáis el software. [Para vosotros] El hardware debe pagarse, pero el software debe compartirse. ¿Quién se preocupa de que los autores obtengan su paga?*

# El nacimiento del software propietario

- *¿Pero qué es esto? Como muchos de los hobistas ya debéis saber, la mayoría de vosotros robáis el software. [Para vosotros] El hardware debe pagarse, pero el software debe compartirse. ¿Quién se preocupa de que los autores obtengan su paga?*

“Carta a los Hobistas” de Bill Gates,

3 de febrero de 1976

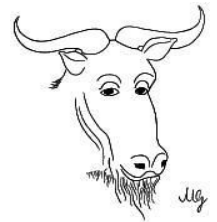
<http://www.blinkenlights.com/classiccmp/gateswhine.html>

# Software propietario

- Se comercializa como un producto manufacturado
- Pero con condiciones que no se da en ningún otro producto o mercado:
  - No se puede modificar, no se puede adaptar
  - El productor impone las condiciones y casos de uso
  - No existen regulaciones aunque existan grandes monopolios

# Nacimiento de GNU

- En 1984 Richard Stallman, del MIT, reacciona y decide crear un sistema completo similar al UNIX: GNU (*GNU is Not Unix*)
- En 1985 crea la *Free Software Foundation* para recoger donaciones y contratar programadores
- El primer producto conocido fue el Emacs
- Uno de los más grandes y complejos es el compilador multiplataforma más sofisticado: gcc



# Derechos de autor, *Copyleft* y GPL

- Dominio público
  - Pueden agregarse restricciones adicionales y dejar de ser libre
- *Copyleft*
  - Es software libre, se mantienen los derechos de autor
  - No permite que se puedan agregar restricciones adicionales a las modificaciones o distribuciones
- GPL (*General Public Licence*)
  - Licencia de distribución que especifica los términos “copyleft” de un programa

# Software Libre vs. Software abierto

- Se creó (1998) cómo una forma de acercar la antigua idea *hacker* a las empresas
- No hay grandes diferencias de estrategia ni de recomendaciones prácticas, sino de principios básicos:
  - Software libre = libertades fundamentales
  - Código abierto = metodologías de desarrollo y colaboración
- Comparten criterios, metodologías y proyectos

# Las tecnologías de la información en las empresas

Costes, inversiones, ¿ventajas competitivas?,  
infraestructura, economía del software

# ¿Quién pudo haber escrito lo siguiente?

- ¿Un defensor del software libre?
- ¿Un competidor de Microsoft?
- ¿Un técnico o científico?
- ¿Lula? ¿Rodríguez Ibarra?

Cada año, las empresas compran más de 100 millones de PCs que en su mayoría reemplazan modelos anteriores. Pero la inmensa mayoría de trabajadores que usan esos PCs sólo trabajan con unas pocas aplicaciones sencillas **-procesador de texto, hoja de cálculos, email, navegador web-**. Estas aplicaciones ya están maduras desde hace años.

...

La mayoría de ese gasto, hay que decir la verdad, **está dirigida por las estrategias de los vendedores**. Los grandes suministradores de hardware y software se han especializado en distribuir planificadamente las nuevas características y capacidades, de forma que obligan a las compañías a comprar nuevos ordenadores, aplicaciones, y equipamiento de redes mucho más frecuentemente de lo que necesitan.

...

Si los vendedores se resisten, las empresas deberían explorar **nuevas soluciones que incluyan al software abierto** y PCs de red minimalistas, aún sacrificando características. Si una compañía necesita evidencia de la cantidad de dinero que podría ahorrarse, basta **echar un vistazo a los márgenes de ganancias de Microsoft**.

# Harvard Business Review

IT Doesn't Matter

Nicholas G. Carr [\*]

Harvard Business Review, May 2003

*Al tiempo que la potencia y ubicuidad de las tecnologías de la información se ha incrementado, su importancia estratégica ha desaparecido. La forma en que afronta las inversiones y gestión de las TI necesitará cambiar radicalmente.*

[\*] Editor de *HBR*. Editor de *The Digital Enterprise*. Articulista en *Financial Times*, *Business 2.0* e *Industry Standard*.

# ¿Quién lo dijo?

*La mayoría de las empresas gastan demasiado en TI y obtienen muy poco retorno.*

# ¿Quién lo dijo?

*La mayoría de las empresas gastan demasiado en TI y obtienen muy poco retorno.*

Larry Ellison

Director y Consejero Delegado de Oracle Inc.

# Infraestructura según HBR

- Las TI se han convertido en tecnologías de infraestructura.
  - Todos deben pagarla para poder hacer negocios pero no provee distinción a nadie.
  - El impacto estratégico de las TI proviene del efecto acumulativo para innovar en las prácticas de negocio.
- Los vendedores están cambiando el modelo de negocio a suscripciones.
- **La tecnología es suficiente, la creatividad es la que escasea.**

# ¿Qué es la infraestructura?

- Ofrecen mucho más valor cuando se comparten que usadas de forma aisladas. -- Nicholas Carr, HBR
- Los estándares ayudan a definir las infraestructuras.
- Los elementos de las infraestructuras son *commodities* (bien de consumo).
- El mercado de las *commodities* es un mercado de competición pura.
- Los mercados de competición pura es el mercado dictado por los consumidores.

# Microeconomía básica

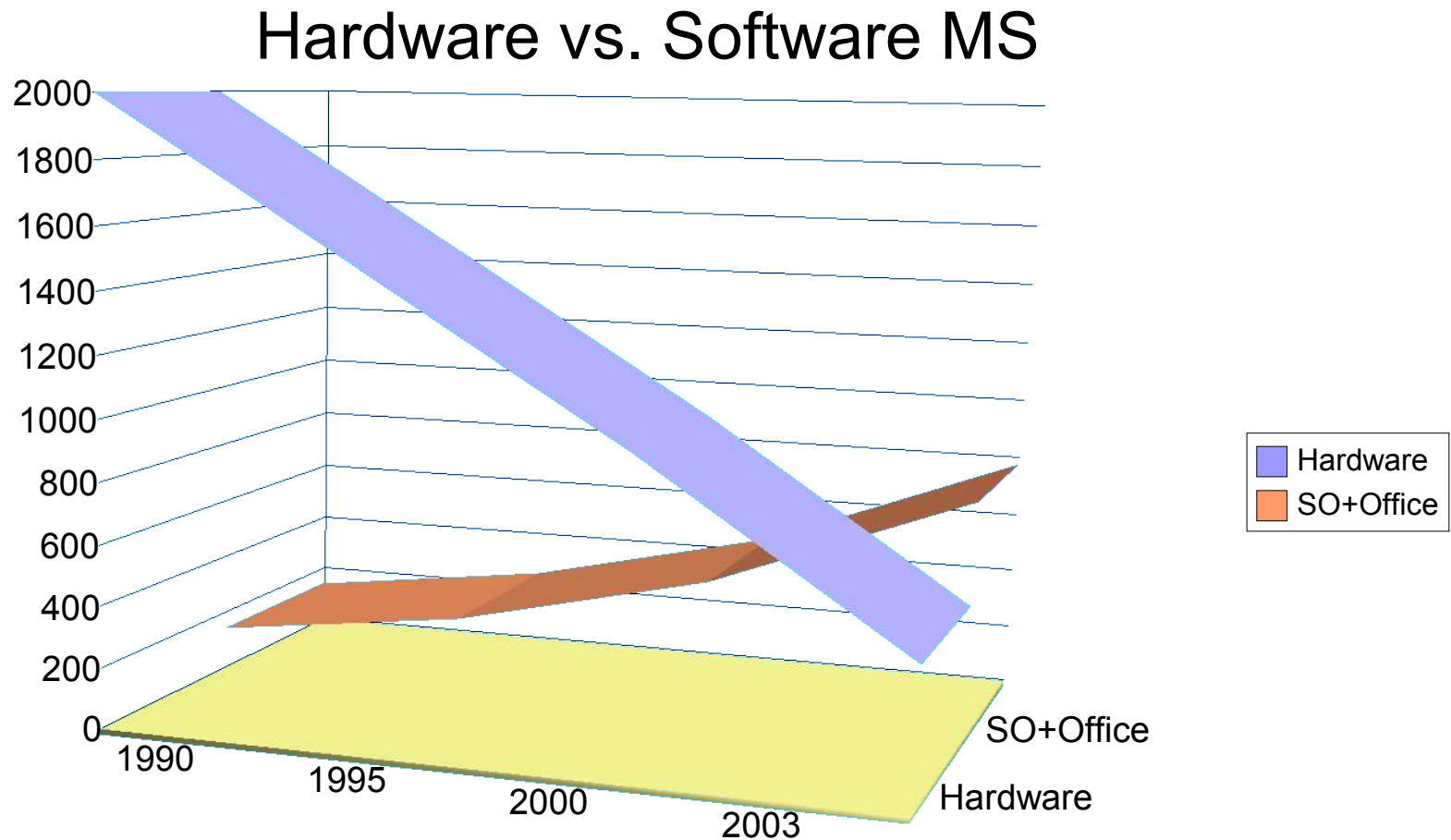
- Producto competitivo. Características similares, si baja uno obliga a bajar a la competencia.
- Producto complementario. Se puede subir el precio si baja el del complementario.
- Producto sustituto. De características similares pero exige una toma de decisión importante.
- Bien de consumo o *commodity*. Los precios están minimizados, muy cercanos a los de producción.

**El mercado del bien de consumo es una pesadilla para los productores**

***Smart companies try to commoditize their products' complements***

**Las compañías inteligentes intentan que sus complementarios sean productos de consumo**

# Evolución de precios



# El hardware es un bien de consumo

- El software es altamente “incremental” y reciclable, con costes marginales ínfimos.
- El hardware es un bien de consumo.
  - Lo ha iniciado IBM al estandarizar los componentes del IBM PC.
  - Los márgenes del hardware son mínimos.
- El software propietario no es un bien de consumo.
  - “Decomoditizar” los estándares y protocolos
    - Hacerlos complejos, especificación incompleta, cambios frecuentes, hacer que contengan propiedad intelectual protegida, agregar valor diferenciador

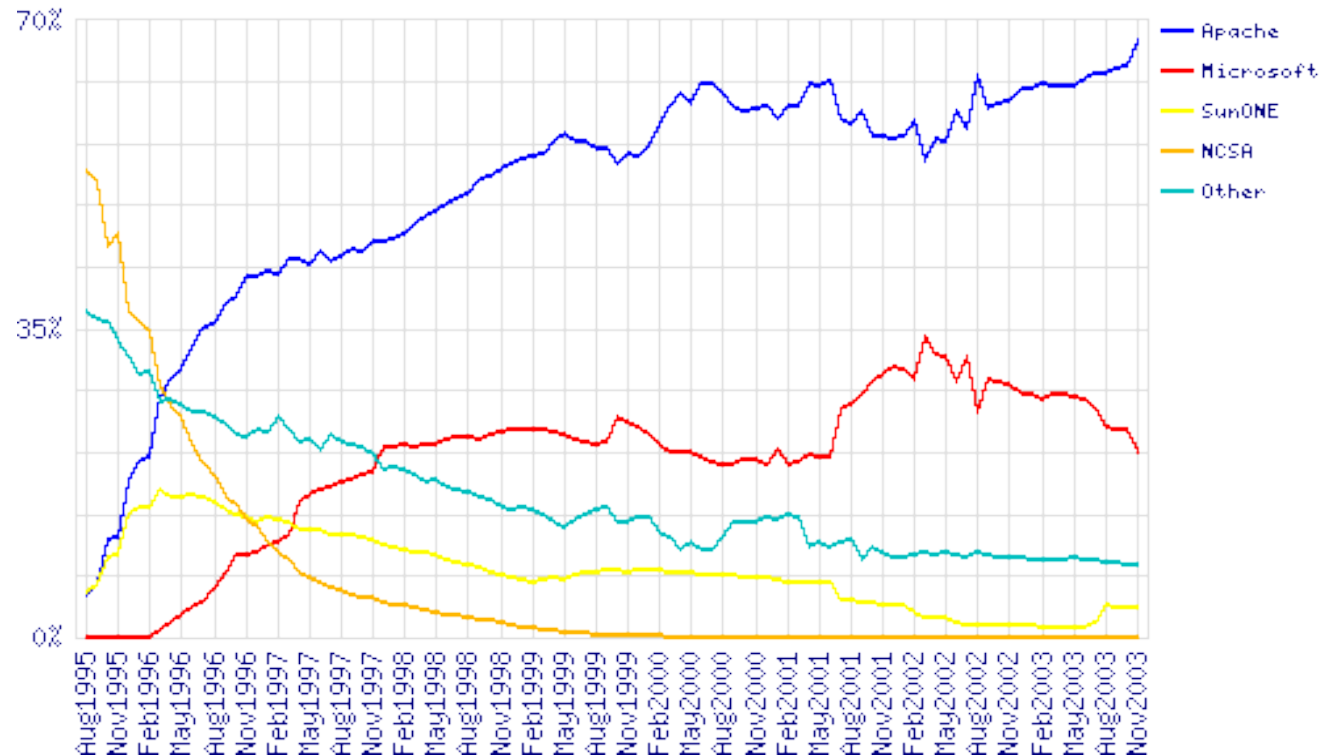
# El software como bien de consumo

- ¿Es el software un bien de consumo?
  - ¿Puede cambiar su sistema operativo sin problemas?
  - El mercado del software propietario no es competitivo, ni eficiente.
- ¿Es o será el software un bien de consumo?
  - ¿Que efecto tienen Internet y los estándares?
- ¿Es el software libre una competencia al software propietario?
- ¿Cuál es el precio de producción del software?

# El SL ha dado la talla

- Toda la infraestructura de las modernas tecnologías de la información ha sido soportada por el software libre:

- dns
- email,
- ftp,
- web,
- samba,
- perl
- ...



# Economía del SL

Sobre la motivación de los programadores y los consumidores

# El SL no es económicamente viable

- Es incompleto, sólo mira la mitad de la ecuación: los suministradores.
- No hace falta dinero para que exista el SL.
- Los consumidores están ahorrando mucho dinero.
- Punto de vista economicista: *dinero ahorrado es dinero ganado*.
- La demanda dirige la cadena de suministro, y pide menos precios.
- Surgirán otro tipos de suministradores.

# Los consumidores matarán a la industria del software

- ¿Deben los consumidores pensar en su propio dinero o en la cadena completa de suministro? ¿Se les puede pedir que compren productos más caros para favorecer a la “cadena de producción”?
- **¿El estado sobre el individuo?** Parece más un argumento de Marx que de Smith.
- Fabricantes de carruajes vs. el tren, o ferrocarril vs. autopistas. La patente Selden y Ford.
- Los consumidores eligen y los suministradores se adaptan o perecen.

# ¿Continuarán contribuyendo sin cobrar dinero?

- No económica no es la única razón.
- Existen otros tipos de transacciones además de *win-win, win-lose, lose-lose: lose-neutral, neutral-neutral, win-neutral*
- Los desarrolladores no perciben que tenga un coste. *Scratch your own itch.*
- Colaboración. *Se pone un ladrillo y se obtiene una casa.*
- La colaboración no tiene suma cero.

# ¿Y los costes de oportunidad?

- Es más problemático poner un programa en el mercado que programarlo.
- La barrera de introducción al mercado de software propietario es muy alta.
- La tasa de éxitos es muy baja.

# ¿No generan ellos mismos pérdidas de sus puestos de trabajos?

- El 95% de los programadores desarrollan software para uso *in-house* y/o personalizado.
  - Ese 95% es exportable a las empresas desarrolladoras.
- El SL no compite contra ese 95% de los programadores, sólo con el 5% restante que vende programas empaquetados.
- En términos muy usados últimamente, es un “daño colateral menor”.

# Economía del SL

El SL en las empresas

# La GPL es viral y pone en riesgo a todo el software de una empresa

- La licencia GPL es como cualquier otra licencia.
- Es una licencia de distribución.
- Está basada en las leyes de propiedad intelectual.
- Es una **autorización** de los autores.
- Si se infringe, se violan las leyes de protección intelectual.
- Hay varias formas de remediar una violación.
- El código sólo puede ser convertido a GPL con la **expresa voluntad** de sus propietarios.

# No existe la comida gratis, no es natural

- ¿No viola alguna ley fundamental?
- En un mercado competitivo puro, los precios de venta deberían ser muy cercanos a los de producción.
- Si el coste de producir es cero, y los costes marginales son cero, el precio cero está justificado.
- Lo no natural es que el software sea parte importante de la infraestructura, pero no sea un bien de consumo ni tenga precios adecuados a sus costes de producción y marginales.

# ¿Será el software una *commodity*?

- Estándares, Internet y el mismo software libre
- Coste marginal con el web es cero.
- El web es una tecnología predominantemente de servidor: HTTP + HTML, igual para SMTP, DNS, LDAP...
  - No hay marcas más “blancas”.
  - La marca del servidor no es visible.
  - La diferenciación no existe.
  - Todos los estándares tienen al menos una implementación en SL.

# El TCO es más alto que el del SP

- ¿Lo dice un informático? La resistencia al cambio.
- ¿Lo dice un economista?
  - El TCO de un Citroen Picasso es muchísimo menor que un autobús Mercedes Benz, ¿porqué la EMT no compra Picassos en vez de costosos autobuses?
  - Lo que importa es el **retorno de la inversión**.
- ¿Lo dice un “productor”? Tienen que justificar el precio no competitivo.
- Los administradores de Linux son más “caros”. Es un problema del sistema educativo.

# Las distribuciones de Linux son caras para las empresas

- No venden el producto, sino servicios.
- Hay muchas opciones. El soporte puede darlo hasta una empresa local.
- Distribución y mantenimiento de un sistema operativo y cientos de aplicaciones:
  - Más de 60.000.000 de líneas de código valoradas en más de 2.000 millones de dólares usando métodos de estimación de software propietario SLOC y COCOMO: 15.000 personas/año, con una duración estimada de 6 años...

# Economía del SL

Desarrollo del SL en las empresas

# Las empresas no desarrollarán de SL si no hay ganancia (I)

Menos del 1% de las empresas desarrolla software. Menos del 0.5% de las empresas ganan dinero vendiéndolo.

- Es muy raro que el coste de programación sea casi cero.
- Linux jamás se hubiese podido desarrollar dentro de una empresa.
- Las empresas requieren recuperar la inversión. Aunque el coste marginal sea cero, deben cargar una cantidad para recuperar los costes de desarrollo.

# Las empresas no desarrollarán de SL si no hay ganancia (II)

- Para que sea posible la venta, el software debe asimilarse al de un producto físico y la replicabilidad infinita debe ser eliminada.
- Se hace a través de la legislación.
- Funciona porque se impone al sistema una escasez artificial.
- ¿Que pasa si el coste de desarrollo pudiese **apuntarse como un gasto, sin necesidad de recuperar la inversión?**

# Las empresas no desarrollarán de SL si no hay ganancia (III)

- No pueden apuntarse grandes gastos sin una recuperación pero el gasto puede ser pequeño
- El software es único: puede ser desarrollado por cientos o miles de individuos.
- El software, como la definición capitalista de “riqueza” puede ser infinito. “Caldero mágico”.
- ¿Será que el modelo de software propietario es equivocado por ineficiente?
  - Se necesita un modelo más “capitalista” y menos “mercantilista” ¿la “mano invisible” de la sociedad?

# ¿Quién dominará el mercado?

- *El SL tendrá su nicho, pero software propietario seguirá dominando el mercado. ¿Es realmente un mercado de bienes de consumo? ¿No está el software propietario más adelantado? ¿Acaso Oracle no es mejor que sus competidoras de SL?*
- Es una falacia. No hace falta ofrecer más funcionalidades que el software propietario. En los estándares e infraestructura, **más es menos.**
- ¿Cuanto pagaría por funcionalidades que nunca usará?

# Si es tan bueno, ¿porqué no se adoptó más rápido?

- El SL es un suplemento del SP, requiere un cambio de mentalidad.
- Eso lleva tiempo, se necesitan ver ejemplos positivos de los *early adopters*.
- Ahorros potenciales y control total sobre la tecnología (y estándares) serán los argumentos más importantes.
- Pero cuando se trata de infraestructura, es decir mercado de competencia pura **el SL ha dado la talla y seguramente lo seguirá haciendo.**

# El SL divulga riqueza, pero no la genera

- Es el argumento típico, pero los analistas indican que las empresas generan más valor mediante el ahorro e inversión muy medida en la tecnología (y ahorro de costes).
- Los procesos y actividades de negocios actuales son ineficientes, por lo tanto se pierde “valor”.
- Los estudios indican que no hay relación directa entre las inversiones tecnológicas y el éxito o posicionamiento estratégico de las empresas (6 de 58).

# Economía del SL

El SL y la “innovación”

# El SL no innova (I)

- Es el argumento repetido hasta la extenuación por los vendedores y directores de comunicación de las empresas de software propietario.
- Hay estudios científicos que demuestran que la mayoría de las invenciones que usamos actualmente es de la década de los 80 y anteriores.
- Ejemplos de SL:
  - Servidores: cern, httpd, apache, CGI, Perl, lenguajes empotrados, bases de datos adaptadas a web, *thin clients* (X-Terminals), administración remota.

# El SL no innova (II)

- Los monopolios innovan más lentamente.
  - Páginas web “tabuladas”, gesturas, bloqueo de popups, antispams y firmas digitales en el correo electrónico...
- Cuanto tiempo ha tardado Microsoft en implementar:
  - Navegador web,
  - Reproductor MP3 como el Winamp (X11amp)
  - Mensajería instantánea (ICQ, Jabber)
  - Vídeo (xanim, RealVideo)

# El SL no innova (III)

- Favorece la innovación:
  - Amazon esta totalmente basada en Perl y Linux.
  - Google es todo Linux (más de 20.000 servidores).
  - TiVO, el primer grabador de vídeo digital doméstico (199/) está basado en Linux
  - Los AP Linksys, comprados por Cisco, es Linux y están considerados los mejores del mundo (hasta por Google)
  - Muchos *setup-box* de TV digital (y la mayoría de las plataformas de desarrollo) son Linux.

# Las patentes dificultan al desarrollo de SL

- Es verdad. Pero la patentes **dificultan cualquier tipo de desarrollo de software.**
- Cualquier programa viola muchas patentes.
- Hubo más innovación en los 80 que en los 90, cuando se relajaron los criterios de patentabilidad.
- Hay estudios científicos, muestran que las patentes no son una actividad complementaria de la I+D.
- El 69% de las patentes de software son de empresas ni informáticas que sólo emplean al 10% de los programadores.

# Economía del SL

## El negocio informático

# ¿Y el negocio informático? (I)

- **Desarrolladores de paquetes propietarios:** cuestión de elección y de mercado.
  - Si hay alguien que desee pagarlo, lo podrá vender. Pero hay que pensar que el software se convertirá en una *commodity*. Seguirán con el problema de la barrera de entrada al mercado.
  - El 75% de los costes del ciclo de vida de un programa son de mantenimiento y para reducir el *gap* del modelo con el mundo real.
  - Con la colaboración se obtiene actualizaciones externas gratuitas (Websphere de IBM).

# ¿Y el negocio informático? (II)

- **Integrador:** es el paraíso. Esta actividad es complementaria al software y hardware. Todos los productos a su disposición, respeta estándares, bajo coste. Facilidad de adaptación.
- **Mantenimiento y servicios.** También es complementaria al software y hardware.
- **Desarrolladores de software libre:** es el mercado nuevo, acompañado de las tareas anteriores.
- **Fabricantes de hardware:** Gran reducción de costes. También es es complementaria al software.

# ¿Y el negocio informático? (III)

- Quizás se llegue a un mercado competitivo y donde cuenta el conocimiento, experiencia y prestigio.
- Como los abogados, contables, arquitectos, ingenieros, médicos, clínicas...
- Todas las profesiones y negocios anteriores se basan en el conocimiento de técnicas o información de dominio público.

# Conclusiones

# Conclusiones (I)

- Las empresas gastan mucho dinero en tecnologías de la información sin obtener los retornos esperados.
- Las tecnologías de la información es la infraestructura (no representan ventajas competitivas “individuales”).
- El SP no es un bien de consumo.
- El SL es un modelo económicamente viable y existe. Seguirá existiendo y creciendo.

# Conclusiones (II)

- El SL está revolucionando la ingeniería informática. No hay explicación formal.
- Se acerca más al modelo científico: publicación, revisión de pares, construcción incremental (la rueda se inventa una sola vez).
- El SL beneficia con reducción de costes a casi todas las empresas, salvo a las que se dedican a producir paquetes de SP, menos de un 0.5% del total de empresas.

# Conclusiones (III)

- En el caso de empresas de personalizaciones, mantenimiento o integración puede significar mayores ganancias al ser su actividad complementarias del hardware y software.
- Ha permitido la creación de nuevos mercados como los de Amazon, Google, TiVO, Yahoo, etc. y la mayoría de los proveedores de Internet.
- Presenta problemas de sustitución a una minoría de empresas, pero abre nuevos mercados y baja las barreras de entrada.

Finalmente el final...