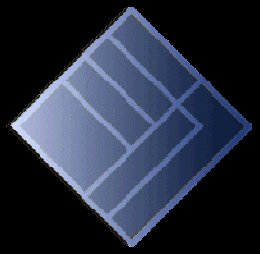
Modelos de Negocio sobre Java, Software Libre y Cloud Computing

Sergio Montoro Ten
knowgate

Objetivos de la sesión

1 ¿Porqué escribir un proyecto en Java?
2 ¿Porqué escribir un proyecto en absoluto?
3 ¿Qué ingredientes debe tener un producto vendible?
4 ¿Qué influencia ejercerá el Cloud Computing?



Historia de knowgate

- Fundada en 1999
- 4 socios fundadores y 4 millones de pesetas
- Foco en productos y servicios para grandes cuentas
- 1 crisis severa tras el estallido de la burbuja .com
- 1 producto estrella en Sw Libre: hipergate CRM
- Estado actual: PyME consolidada (14 pax x 0,9M€ Fact.)

orange™ rtve YELL. INFORMÁTICA El Corte Inglés

BMW SEAT IBERIA

¿Por qué escribir un proyecto en Java? (y no en cualquier otro lenguaje)

1 ¿Qué necesita el cliente? ¿Qué infraestructura tiene?
2 ¿Con qué partners poderosos se puede contar?
3 ¿Cómo es de importante la productividad?
4 ¿Cómo es de importante la compatibilidad?
5 ¿Cómo es de importante el rendimiento¿
6 ¿Cómo es de importante la disponibilidad de mano de obra?

¿Qué es un Modelo de Negocio?

Es un documento que explica cómo ganar clientes y mantenerlos.

Dice quién pagará qué cantidad de dinero por cual concepto y en qué momento del tiempo.

* La mayoría de las empresas de software se montan para venderlas y su valor es aproximadamente igual al número y rentabilidad de clientes que tengan.

Cómo se valora un Modelo de Negocio

1 Potencial bruto de ingresos

2 Cantidad de inversión que requiere

3 Grado de competencia presente y futura

4 Innovación en el propio modelo

5 Facilidad/dificultad para vender

6 Riesgos tecnológicos

7 Riesgos legales

Entonces, el Sw Libre NO es un Modelo de Negocio, es:

1º) Una forma de licencia de propiedad intelectual.

2º) Una forma de compartir y fomentar la innovación y los trabajos derivados.

3º) Una forma de distribuir software.

Valor de Uso vs. Valor de Venta

valor de uso: beneficios que el cliente obtiene con la utilización del programa;

valor de venta:dinero que estaría dispuesto a pagar por ese mismo software.

¿Dónde está el valor del software?

Sólo una pequeña fracción del precio que el cliente paga por un software está en el propio código. El resto es un valor que proviene de la posición del producto en el mercado y de la capacidad del proveedor para dar servicio.

En otras palabras el software es una industria de servicios operando bajo la persistente pero infundada ilusión que es una industria de manufacturas.

¿Por qué apareció el Software Libre?

La aparición de modelos de negocio Open Source no está necesariamente vinculada a la popularización de las licencias libres.

Dichos modelos aparecen por un **desequilibrio** en la relación de poder entre el proveedor y el cliente.

Un sistema está en equilibrio cuando ninguno de los incumbentes puede ganar nada unilateralmente cambiando su estrategia.

El el caso del Sw Privativo los usuarios pueden ganar más cambiando su estrategia de compra y por eso se produce un movimiento de cambio.

¿Cual es el problema del Sw Privativo?

Para un fabricante, básicamente es igual de caro crear el software que mantenerlo vivo.

Sin embargo, en el modelo privativo clásico, se carga un alto precio por la licencia al principio y (en teoría) menos precio por las actualizaciones.

El modelo es insostenible a menos que:

a) las ventas crezcan todos los años de modo que las licencias cubran los costes de mantenimiento.

o **b)** se cobre prácticamente lo mismo por la licencia que por las actualizaciones, -lo cual suele ser visto como algo abusivo e ilógico por parte del cliente-.

Modelo de valor de venta indirectos

1 Posicionamiento en el mercado
2 Asegurar el futuro
3 Regalar la receta, y abrir un restaurante
4 Integrar componentes *commodities*
5 Open Core / Open Source Comercial
6 Liberar el futuro y vender el presente
7 Liberar el software y vender la marca
8 Liberar el software y vender contenidos
9 Ofrecer licencia duales

Modelo de valor de venta indirectos I

1 Posicionamiento en el mercado

Usar un producto libre para vender otro propietario

2 Asegurar el futuro

Usar un producto libre para vender otra cosa que no es software

3 Regalar la receta, y abrir un restaurante

Vender servicios y actualizaciones

4 Integrar componentes *commodities*

Vender integración de otros componentes

5 Accesorizar

Vender accesorios y piezas de recambio

Modelo de valor de venta indirectos II

6 Liberar el futuro y vender el presente

Ofrecer licencias con temporalidad

7 Liberar el software y vender la marca

Capitalizar la marca

8 Liberar el software y vender contenidos

Vender contenidos para un software libre

9 Ofrecer licencia duales

Vender una exención de las obligaciones de la GPL

10 Open Source Comercial

Vender lo que no es

Ingredientes clave para vender un producto de Software Libre

1 Debe ser una solución completa.
2 Debe contener una propuesta de valor concreta para el cliente final.
3 Debe haber un modelo de negocio para los revendedores e intermediarios.
4 Debe ser más barato de adoptar que de fabricar.
5 Debe dirigirse al nicho de mercado apropiado.
6 Debe contar con el apoyo de otros fabricantes.
7 Debe ser lo más global posible.
8 El marketing es más importante que la tecnología.
9 El canal es más importante que la tecnología.

4 grandes ideas ingénuas

1 El Gran Tonto

Si fabricas algo, alguien vendrá y te lo comprará por más dinero del que te costó.

2 El Gran Creyente

Si fabricas una parte de algo grande alguien vendrá y completará el resto.

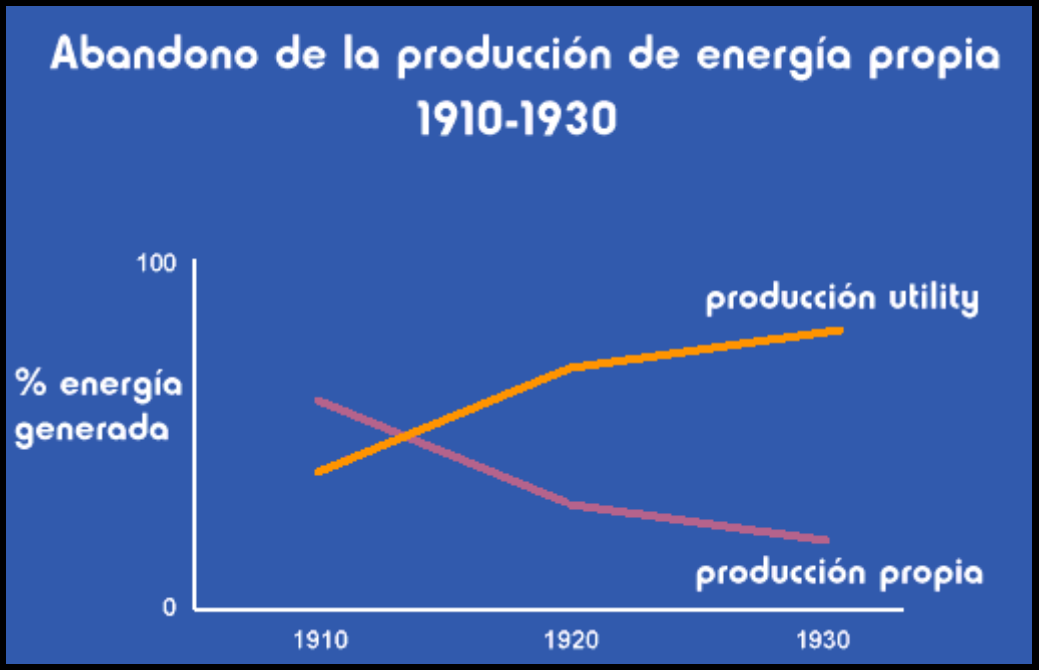
3 El Gran Ingeniero

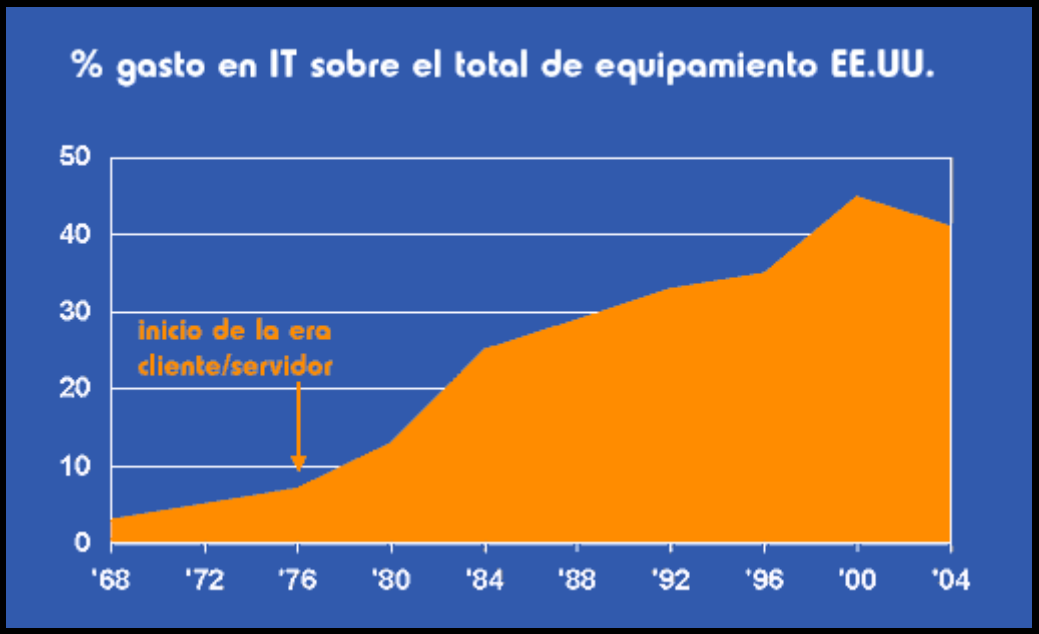
Si fabricas una mejor trampa para ratones la gente empezará a ir a tu ferretería.

4 El Gran Héroe

Con nuestro gran coraje venceremos disparando de frente contra los Casacas Rojas

Cloud/Utility Computing





Redshift

El Utility Computing es inevitable porque la demanda de ciclos de CPU y espacio en disco seguirá creciendo de forma exponencial en los próximos años y no sería eficiente atenderla con el modelo actual de computación.

Pila de nuevas tendencias

Utility Computing
Suministro de recursos computacionales (procesamiento y almacenamiento) como un servicio medido similar a las utilidades públicas tradicionales: electricidad, agua, gas.

Software as a Service (SaaS)
Modelo de distribución de software en donde se provee el servicio de mantenimiento, operación diaria, y soporte del software usado por el cliente.

Cloud Computing
Tecnología para servicios de computación online, de modo que los usuarios puedan acceder a los servicios sin conocimientos en la gestión de los recursos.

Grid Computing
Forma de computación distribuida, que permite utilizar de forma coordinada todo tipo de recursos que no están sujetos a un control centralizado.

Virtualización
Separación entre los recursos físicos y lógicos en la computadora de modo que el software no tenga conocimiento directo de la arquitectura hardware que lo soporta.

La relación entre Java, el Sw Libre y el Cloud Computing

Java: Para que el Utility Computing se convierta en una realidad se requiere un tipo de software altamente escalable, virtualizable, monitorizable, estandarizable y de mantenimiento automatizado.

Sw Libre: El Software Libre es prácticamente el único tipo de software que puede emplearse en utility computing debido a que no incrementa los costes de licencia según aumenta el número de usuarios.