Participando en proyectos de Software Libre: Aspectos técnicos y sociales

Gunnar Wolf — gwolf@gwolf.org
http://www.gwolf.org/soft/qa_soft_libre

Instituto de Investigaciones Económicas, UNAM
Desarrollador del proyecto Debian

3er Día Linux y de Software Libre
ITESM, Cuernavaca, 24 de abril, 2007

Contenidos

Primer acercamiento

¿Qué es el Software Libre?

- Un movimiento social que busca corregir una aberración histórica
- Miles de voluntarios que rompen toda predicción económica
- Un movimiento eminentemente social, con efectos técnicos como primer resultado
- Un nuevo modelo de producción de software
- Un esquema justo de licenciamiento de la propiedad intelectual

Sí, pero... ¿Qué es el software libre?

- El mundo del cómputo hasta los 70 seguía otras reglas
  - El negocio está en vender hardware — El software requerido va incluído
  - Es de esperarse que los usuarios quieran modificar tanto hardware como software
  - Los sistemas se venden con esquemas y código fuente completos
- La revolución de las PCs nos trajo un licenciamiento injusto e ilógico
  - Al pagar por un programa, no lo compramos — Sólo adquirimos una licencia de uso bajo ciertas condiciones
  - Lo compramos tal y como está — Es imposible adecuarlo a nuestras necesidades específicas
  - La compañía dueña del código no da garantía alguna sobre de él
  - ¡Pero no recibimos siquiera el derecho de corregir problemas!

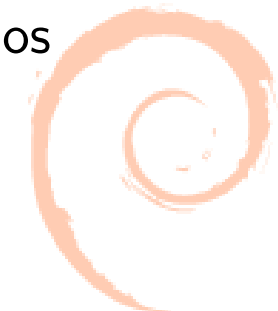
Sí, pero... ¿Qué es el software libre?

- El software libre básicamente nos devuelve la propiedad y los derechos
  - Con el software libre recuperamos el control de nuestra propiedad
  - Podemos adecuar el software a nuestras necesidades
  - Podemos corregir los problemas que presente
  - Podemos compartir nuestro desarrollo con quien lo necesite

Aterrizando

Un programa es considerado software libre si su licencia nos garantiza:

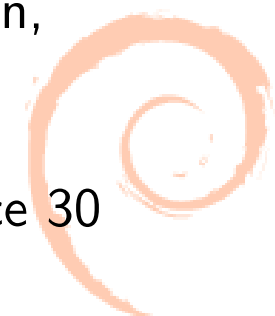
- Libertad de uso
  - Podemos usar el programa para el propósito que deseemos
- Libertad de aprendizaje
  - Podemos aprender cómo está hecho el programa
  - Tenemos acceso a su código fuente
- Libertad de modificación
  - Podemos adecuar el programa a nuestras necesidades modificando su código
  - Podemos incluir partes de su código en nuestros desarrollos
- Libertad de redistribución
  - Podemos compartir el programa con otras personas
  - Podemos compartir el código fuente con otras personas
    - En su estado original o modificado



Aclarando

El software libre simplemente es diferente.

- Rompe con la lógica de trabajo en la industria
  - Plantea hasta cierto punto la vuelta a una forma de trabajo académica
- Ha creado fuertes presiones sobre una industria basada en principios errados
- Va a modificar radicalmente la manera de trabajar de esta industria
  - Para concentrarse en la venta de servicios (personalización, administración, etc.)
  - Adecuaciones a las necesidades específicas del usuario
  - Vuelta a un modelo más sano, como lo que teníamos hace 30 años



Aclarando

Por increíble que parezca, para muchos el Software Libre se ha convertido de un movimiento social, político y técnico en una religión.

San Ignucio de la Iglesia de Emacs

... Y a todo esto, ¿por qué participar?

- Vamos a dejar esto para un poco más terde
- ...Hablemos primero de cómo funciona el desarrollo, de cómo funciona el desarrollo y cómo se estructuran las comunidades
- Seguramente ustedes irán encontrando suficientes razones para interesarse

Contenidos

Crisis en la... ¿*industria* del software?

- Es común oir hablar de una crisis en la industria del software. Pero el problema no es, como veremos, el que haya una *crisis*, sino el que veamos al desarrollo de software como una *industria*.
- El desarrollo de software puede —y debe— crear y desarrollar fuentes de trabajo. Sin embargo, el equiparar el desarrollo de software con el trabajo industrial, con líneas de producción y procesos predecibles y repetibles es claramente un error

Desarrollo de un proyecto

Dentro de la industriparalelo de a tradicional del software tienden a observarse metodologías, divisiones y tiempos, rígidos y claros, para las diferentes etapas

- No hay una sóla concepción de cómo debe ser el proceso ideal, pero todo proyecto tiene claro y definido el suyo
- La impredictibilidad y el caos son vistos como los mayores enemigos
- Proyección de funcionalidad (“roadmaps” del proyecto) a mediano y largo plazo

La presión del tiempo al mercado

La crisis de la industria del software deriva precisamente de que es visto como una industria.

- Necesidad de vender los productos en un tiempo competitivo
  - Resultado: Sistemas menos maduros, más vulnerables, con cantidades tremendas de fallos conocidos (dicen los que saben, decenas de miles en Windows XP)
  - Por lo visto, Microsoft comprendió *algo* de sus graves errores — Su nuevo sistema *Vista* salió al mercado con un retraso de cinco años, y sin funcionalidad novedosa por sobre su competencia

La presión del tiempo al mercado

Competencia en vez de cooperación

- En nombre de los secretos industriales, patentes sobre algoritmos y, en general, la propiedad intelectual, la computación como disciplina científica está atascada desde hace más de 20 años
- Especialmente durante los 80 y 90 vimos una y otra reimplementación de las mismas ideas de diferentes maneras, con interfaces completamente deiferentes, en vez de cooperación para el desarrollo de nuevas tecnologías
- ¿Son esas buenas noticias? ¡No! Si hoy en día no hay más trabajo tirado a la basura no es por que haya competencia, sino porque hay un potente monopolio. En las áreas aún no conquistadas por éste, sigue habiendo duplicación innecesaria de esfuerzos.

La mal llamada *propiedad intelectual*

- Toda empresa de desarrollo de software tiene que pagar grandes despachos de abogados
- Toda empresa de desarrollo de software aparentemente debe tener una sólida cartera de patentes sobre algoritmos
  - Principalmente como defensa
  - ...Pero también para hacerse de fondos
  - Hay empresas que se dedican exclusivamente a la acumulación de patentes, actuando como *parásitos* en su entorno
- Potenciando la emergencia de oligopolios/monopolios
  - Las pequeñas empresas sencillamente no tienen manera de competir con las grandes, terminan siendo absorbidas o quebrando
  - Y no es culpa de Microsoft — El modelo mismo fomenta la creación de monopolios

Los engranes de la maquinaria (o cómo insultar al cliente sin perder sus cheques)

Con el paso de los años, el cliente ha ido pasando cada vez más a un papel de menor importancia

- Comenzando con licencias excesivamente restrictivas
  - Puede ponerse en duda su legalidad como contratos — Yo no firmé en ningún lado... ¿Licencias desnudas?
  - Me prohiben cada vez más acciones
  - Limitan por completo mi privacidad, requiriendo que expresamente renuncie a ella
    - ¿Alguna vez han leído un EULA/CLUF?
- El Impuesto Microsoft
  - No puedo elegir una computadora de marca sin que incluya Windows. ¿Por qué?

Los engranes de la maquinaria (o cómo insultar al cliente sin perder sus cheques)

- Imposibilidad de recibir soporte técnico
  - Al pagar la licencia del programa, expresamente acepto que carece de garantía — Estoy dando dinero para recibir un permiso de uso (no estoy comprando nada), sin garantía de que sirva de algo
  - Para instalaciones grandes, puedo pagar (en cientos de dólares) para abrir un ticket de soporte, sin garantía de que me resuelvan el problema (o de que intenten hacerlo)

¿Debemos seguir los mandatos de la visión empresarial?

El cómputo históricamente siempre ha tenido su mayor impulso en la academia

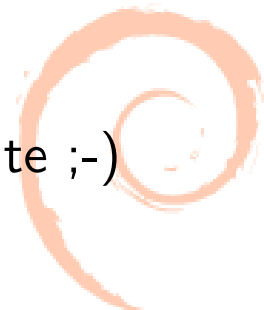
- Hemos seguido ciegamente las tendencias del mercado
- Prácticamente no ha habido verdaderas inovaciones en el cómputo en las últimas dos décadas
- Las prácticas actuales de desarrollo por parte de la industria son ineficientes y caen en todo tipo de vicios
  - Les salió más caro el caldo que las albóndigas

¿Por qué insistir en copiar un modelo inferior?

...Pero el término *industria* suena tan glamoroso...

En ciertos círculos, los jóvenes son formados buscando crear en ellos una *visión empresarial*. ¿Qué tan real es hablar de la industria?

- En el desarrollo de sistemas más aún que en muchas otras ramas de la economía, la proporción de empresas fracasadas es terriblemente alta frente a la de empresas exitosas
- La diversidad en la oferta de productos de software es demasiado baja
  - ...Y el costo de reproducción es casi cero
  - ...No hay una cultura de respeto a derechos de autor
  - ...La competencia de monopolios efectivos es atroz
- La óptica de venta de *productos* está equivocada
  - ...Pero vamos poco a poco — Ese tema viene más adelante ;-)



¿Y cómo es el Software Libre? ¿No lo impulsa la misma *industria*?

- La mayor parte de los desarrollos libre siguen esquemas de desarrollo más cercanos al *académico*
  - El conocimiento generado entra a formar parte del *común* al cual todos los interesados tenemos acceso pleno
  - La recompensa directa que recibe el autor es el crédito a su trabajo y el reconocimiento/prestigio que de éste emana
- Hay muchos y muy importantes proyectos de Software Libre cuyo desarrollo *es* visto con una lógica de industria
  - Mi crítica va también encaminada a ellos
  - Sin embargo, aunque son muchos y prominentes, no son ni la generalidad de los proyectos, ni aquellos con los que con mayor probabilidad colaboremos directamente

Y qué... ¿se supone que debo vivir de comer aire?

Muy bonito, muy bonito... ¿pero cómo encaja mi desarrollo profesional o el de mi empresa en todo esto?

Cambio de la orientación de la oferta / del modelo de negocios

Una empresa de desarrollo de software debería orientarse a vender *servicios*, no productos.

- Vender experiencia en la adecuación de soluciones disponibles universalmente
- Vender administración de un servicio requerido frecuentemente por los clientes — Sobre una base perteneciente a los *comunes*
- ...Deja que tu trabajo te recomiende. ¡Y deja que tu trabajo se esparza por sí mismo!

Contenidos

Modelo general de desarrollo

Cada comunidad de desarrolladores ha llevado la filosofía de desarrollo del sistema en que creció a su modo general de trabajo y colaboración
Es divertido y hasta chistoso comparar lo similares que son, a diferentes niveles, los proyectos (chicos y grandes), las comunidades de desarrolladores y los usuarios del lado libre y del lado propietario

- Independencia/interdependencia entre componentes
- Relación con los pares
- Presunción de capacidades de quien nos sigue en la cadena

Modelo de crecimiento orgánico

El modelo de desarrollo que seguimos puede ser visto como el crecimiento orgánico — tiene una gran semejanza con las células de un ser vivo.

- Los diferentes proyectos tienden a ser tan pequeños e independientes como sea posible, con una interfaz consistente
  - Es fácil reemplazar a uno por otro
  - Diferentes proyectos implementando la misma funcionalidad no compiten — avanzan en conjunto
  - Comparten recursos vitales — a los desarrolladores y a su experiencia
- La redundancia no duele
  - Hay muchos proyectos implementando la misma funcionalidad
  - Selección natural

Tratar al usuario como a un colega

A diferencia del mundo del software propietario, en el Software Libre todo usuario es potencialmente un colega

- Ponemos a su disposición tanta documentación como sea posible
- Esperamos reportes de fallos por parte de nuestros usuarios
  - Posiblemente incluso correcciones / mejorías
- Los desarrolladores responden directamente a las dudas/inquietudes de los usuarios
- Para muchos, nuestra meta es crear más desarrolladores
  - ¡En serio no es difícil! Acérquense y verán

Aporte por sí sólo de estos puntos al control de calidad

Las características antes presentadas por sí sólas ya definen un buen nivel de control de calidad

- Las interfaces al programador (APIs) son estables, definidas y bien documentadas
  - Tenemos la conciencia de que proyectos de terceros pueden depender del nuestro
  - No existe ventaja competitiva de esconder la información
    - Aún más, no es posible esconderla
- El código es más claro, mejor comentado y documentado
  - Las tres virtudes del programador: Flojera, impaciencia, vanidad

Características de los desarrolladores

Para entender la naturaleza de nuestro movimiento, hay que entender a sus miembros...

- Dispersión geográfica
  - Mucho menor ceguera cultural
  - Internacionalización de los proyectos
- Alta proporcion de voluntarios
  - Para bien y para mal
  - Muchos proyectos nacen y atraen desarrolladores por ser una idea interesante
- Potencialmente miles de colaboradores para cualquier proyecto
  - Con suficientes ojos todos los fallos se vuelven obvios
    - Mitos y realidades
  - Diferentes maneras de pensar, diferentes prioridades van enriqueciendo a los proyectos

Características de los proyectos

Y como resultado de las características sociales, encontramos claramente ciertas características técnicas

- Proyectos acotados a cumplir sus objetivos
  - Con mucho menos distracciones
  - Con libertad para definir sus prioridades
  - Ambición focalizada — El proyecto no intenta llegar más allá de lo que realmente busca
- Proyectos apegados a estándares
  - Para permitir la reutilización
  - Para facilitar la comunicación sobre red en ambientes heterogéneos

Características de los proyectos

- Retroalimentación de la comunidad
  - Los usuarios se acostumbran a la posibilidad real de interactuar directamente con los desarrolladores
  - Los proyectos reciben y usualmente incorporan las mejorías requeridas por sus diferentes grupos de usuarios
  - Es más fácil que aparezcan a tiempo los problemas derivados de situaciones frontera o no contempladas

Problemáticas inherentes al Software Libre

El Software Libre también presenta importantes dificultades y retos que superar

- Coordinación de proyectos geográficamente dispersos
  - Zonas horarias
  - Los desarrolladores pueden no tener un idioma común — Es fundamental poder leer, escribir, discutir y desarrollar en ingles, nos guste o no, lingua franca en nuestro campo
- Diferentes puntos de vista entre los desarrolladores: ¿Qué pasa cuando diferentes desarrolladores tienen visiones incompatibles?
  - El convencimiento, aunque sea a regañadientes
  - La salida de uno de ellos (o un grupo) del proyecto
  - La división del proyecto en varios

Problemáticas inherentes al Software Libre

- Problemas derivados del trabajo voluntario
  - El proyecto ya dejó de ser divertido
  - Ya no tengo tiempo
  - Prefiero desarrollar nueva funcionalidad a corregir los problemas existentes
  - No me gustan tus ideas
  - Cuando el dinero entra en juego...
  - Contratos con/sin exclusividad
  - ...

Requisitos de comunicación y colaboración a distancia

- ¿Cómo compartir la información entre los desarrolladores?
  - ¿Cómo compartir el código?
    - ¿Cómo pueden personas en diferentes continentes trabajar sobre el mismo archivo?
  - ¿Cómo dar seguimiento a las inquietudes de los usuarios?
  - ¿Cómo se va a comunicar de manera eficiente un grupo de desarrolladores?
  - ¿Cómo se van a manejar los asuntos privados del proyecto?
    - Manejo de recursos materiales
    - Aletras de seguridad
    - Asuntos organizativos

Contenidos

Depósito de código

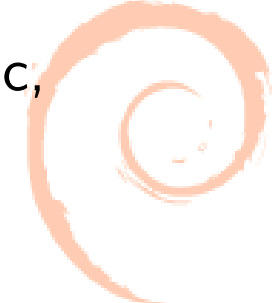
Un proyecto colaborativo, especialmente si es geográficamente disperso (y más aún si tiene decenas de participantes o más), requiere de un depósito de código

- Operaciones básicas
  - Sincronizar cambios en el código
  - Permitir el trabajo sobre el mismo archivo, asistir para solucionar problemas que requieran intervención humana
  - Guardar las diferentes versiones y ramas que ha pasado el proyecto
  - Facilitar la distribución
  - Permitir el *trabajo desconectado*
  - Ramas locales
- Herramientas principales: CVS, Subversion, Git, Bzr, Arch, Mercurial, ...

Seguimiento de fallos (BTS — Bug Tracking System)

Un sistema que permita a desarrolladores y usuarios del proyecto reportar y dar seguimiento a fallas y otras solicitudes

- Que permita dar seguimiento al estado e importancia de cada bug y de cada componente del sistema
- Que permita catalogar/categorizar los diferentes fallos por estado, severidad, categoría, etc.
- Que sea amigable con el usuario final, robusto para el desarrollador
- Que maneje parches, listas de deseos, versiones, ...
- Principales herramientas: Bugzilla, Debbugs, GForge, Trac, muchos otros



Áreas de discusión

Es indispensable contar con espacios donde se pueda discutir con tranquilidad y a profundidad respecto al proyecto. Si bien hay varias alternativas, la principal con mucho son las listas de correo

- Características
  - Facilidad para alta/baja de usuarios
  - Archivo histórico de mensajes enviados
  - Opción para hacer la lista abierta, cerrada, censurada, irrestricta, etc.
- Principales herramientas: Mailman, Majordomo, ezmlm, ...

Herramientas de plática / comunicacizon directa

Aunque ha perdido mucha popularidad entre el público en general frente a los IMs (Yahoo, MSN, ICQ, etc.), entre los desarrolladores el medio inmediato favorito es por mucho el IRC.

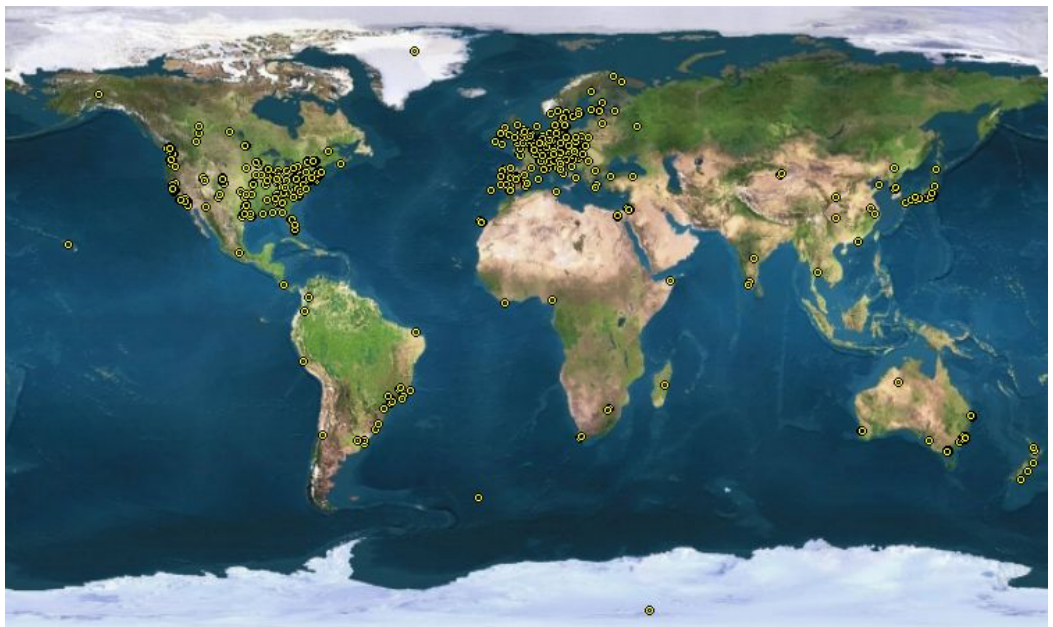
- Plataformas abiertas, no requieren clientes específicos
- Conversaciones uno a uno o muchos a muchos
- Capacidad de archivar fácilmente las conversaciones
- Facilidad de *jugar* con el sistema (bots, scripts etc.)
- Redes públicas que no requieran registro para emplearlas

Contenidos

Presentación general del proyecto

¿Qué es el Proyecto Debian?

- Desarrollo de una distribución de Linux (y otros SOs)
- Un proyecto de integración — *un bazar de catedrales*
- Más de 15,000 paquetes binarios
- Alrededor de 1000 desarrolladores oficiales en todo el mundo



Un vistazo a Debian

¿Cómo logramos trabajar en conjunto?

- Proyecto completamente voluntario
  - Aunque base para el trabajo de empresas (Progeny, Ubuntu, Libranet, Linspire...)
  - Algunos desarrolladores son empleados precisamente para contribuir con Debian (HP, IBM, Nokia, ...)
- El trabajo se basa en ciertos documentos base
  - Directrices del Software Libre de Debian (DFSG) y Contrato Social (SC)
    - Qué consideramos que es Software Libre (y por tanto podemos incluir en Debian)
    - Cuáles son nuestras prioridades
    - A qué nos comprometemos ante la comunidad
  - Políticas
    - Cómo hacemos las cosas
    - Qué debe ir en qué parte del sistema
    - Cómo podemos hacer que nuestros paquetes interoperen correctamente

Ramas de Debian

En Debian se manejan tres ramas o versiones de manera simultánea

- 1- Estable
  - Con las características de estabilidad antes mencionadas
  - Se recomienda a todo usuario
  - La liberación de una nueva versión estable se lleva a cabo aproximadamente cada dos años
- 3- Inestable
  - La rama de desarrollo, que siguen todos los desarrolladores, y se recomienda sólo a quien quiera participar en el desarrollo
  - Alto volúmen de movimiento
  - Los programas individuales que entran a Inestable deben ser estables (son candidatos a entrar a la versión estable)
  - De tiempo en tiempo, algo se rompe
- 2- Pruebas
  - Tras cierto tiempo que una versión de un paquete vive en Inestable y no recibe reportes de fallos mayores, entra a Pruebas

Refrescando estable

Además de las tres ramas oficiales, hay importantes depósitos extraoficiales mantenidos por miembros del proyecto — e incluso por gente externa.

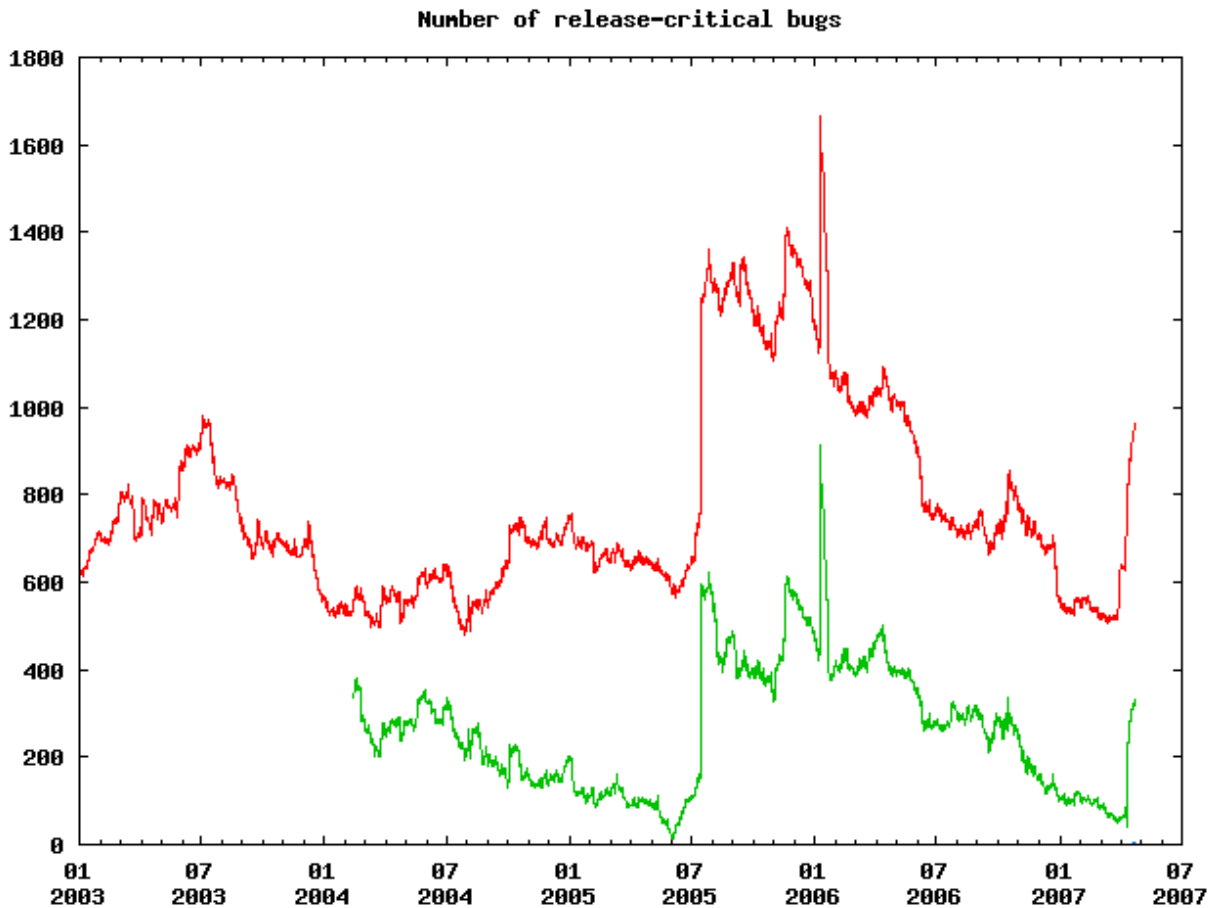
- volatile.debian.net
  - Para software cuya naturaleza hace impráctico el manejo de versiones estables con un ciclo de vida largo (p.ej. herramientas de monitoreo de seguridad, antivirus, antispam, etc.)
- backports.org
  - Versiones actualizadas de paquetes existentes en la rama estable, mantenidas por desarrolladores del proyecto
- apt-get.org
  - Depósitos de software empaquetado para Debian
  - Mantenidos por cualquier persona
  - Aquí podemos encontrar software no empaquetable para Debian por razones de licenciamiento, de inmadurez, etc.

El sistema de seguimiento de fallos (BTS)

La principal herramienta de Debian para el control de calidad es su sistema de seguimiento de fallos

- Sistema completamente público, por estatuto del proyecto (SC, DFSG)
  - Se puede consultar en Web (http://bugs.debian.org/, http://bugs.debian.org/¡numero_de_fallo¿)
  - Se manipula por correo (submit@bugs.debian.org, control@bugs.debian.org, ¡numero_de_fallo¿@bugs.debian.org)
  - Actualmente hay más de 275,000 fallos registrados (unos 26,000 abiertos) desde 1997
- Clasificación de fallos
  - Por paquete
  - Por mantenedor
  - Por severidad (RC, grave, importante, normal, deseos)
- Paquetes virtuales — Principalmente WNPP y ftp.debian.org

BTS — Número de bugs críticos contra la rama inestable (y de pruebas)



Comunicación entre desarrolladores

Debian es un proyecto que se basa muy fuertemente en la comunicación, y cuenta con una gran cantidad de canales

- Listas de correo
  - Varias decenas de listas
  - De todos los niveles (desde debian-user hasta listas muy específicas a arquitecturas, organización, etc.)
  - Abiertas y moderadas
  - Todos los desarrolladores están al menos en debian-devel-announce
- Aliases específicos de correo
  - Para dar con el responsable de un paquete - ¡paquete¿@packages.debian.org
  - Aliases para los roles

Comunicación entre desarrolladores

- IRC
  - Medio principal de comunicación inmediata entre desarrolladores
  - Diferentes canales en irc.debian.org (=¿ irc.freenode.net) - debian, debian-es, debian-devel, debian-boot, etc.
- Reuniones de trabajo
  - Cada que son posibles, sean enfocadas a una necesidad o generales para el proyecto
  - Una reunión principal al año (Debconf)
  - Varias reuniones locales (Oldenburg, debconf-es, Linuxtag, d-i, etc.), oficiales o extraoficiales

El equipo QA

Es necesario un equipo que se encargue de vigilar la calidad general. Sus principales actividades son:

- Mantener a los paquetes huérfanos
- Dar seguimiento a los bugs sobre WNPP
- Construir una y otra vez todos los paquetes en todas las arquitecturas
  - Para verificar que las dependencias sigan correctas
- Busca a los mantenedores aparentemente inactivos (MIA)
- Organizar fiestas de aplastar fallos (Bug Squishing Parties, BSPs)
- Realizar pruebas automáticas (de funcionalidad, actualizabilidad, etc.) sobre paquetes aleatorios

Publicando nuevas versiones

Debian muchas veces lleva a niveles casi absurdos su control de calidad

- Para publicar una nueva versión, requerimos llevar a cero el número de fallos graves
- En un proyeto del tamaño de Debian, la frecuencia de publicación ha ido decreciendo con cada nueva versión
  - La última versión tuvo una gestación de tres años, la actual se acerca ya a los dos
  - ¿Principal causa? La cantidad de gente, y las diferentes prioridades de cada uno de ellos
  - Una de las causas del nacimiento de proyectos como Ubuntu, Componentized, etc.
  - Es un punto de constante polémica en el proyecto
- Es importante tener en cuenta los hábitos de los usuarios fuera de nuestra comunidad
  - Los linuxeros tendemos a buscar lo más nuevo y lo último
  - Los usuarios en general buscan un bajo volúmen de cambios — ¿Cuánta gente conocen que sigue usando Windows 98?

Contenidos

¿Dudas?

¡Gracias!

Gunnar Wolf — gwolf@gwolf.org
http://www.gwolf.org/soft/qa_soft_libre