El control de calidad en proyectos de Software Libre

Gunnar Wolf - gwolf@gwolf.org
Desarrollador del Proyecto Debian
Instituto de Investigaciones Económicas - UNAM
http://www.gwolf.org/soft/qa_soft_libre

El control de calidad en proyectos de Software Libre

1. ¿Qué es Software Libre?
2. Comprendiendo la crisis en la industria del software
3. Características del desarrollo en proyectos de Software Libre
4. Herramientas de desarrollo colaborativo distribuído
5. Un ejemplo: coordinación y control de calidad en el proyecto Debian

¿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

1. ¿Software libre?

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!
- 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

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

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
- Se ha convertido en una religión

El control de calidad en proyectos de Software Libre

1. ¿Qué es Software Libre?
2. Comprendiendo la crisis en la industria del software
3. Características del desarrollo en proyectos de Software Libre
4. Herramientas de desarrollo colaborativo distribuído
5. Un ejemplo: coordinación y control de calidad en el proyecto Debian

Dentro de la industria 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 crisis de la industria del software deriva precisamente de que se ha convertido en una industria.

- Necesidad de entregar 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)
- 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
- La propiedad intelectual
  - Toda empresa de desarrollo de software tiene que pagar grandes despachos de abogados
  - Toda empresa de desarrollo de software debe tener una sólida cartera de patentes sobre algoritmos
- 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

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 - Yo no firmé en ningún lado...
  - Me prohiben cada vez más acciones
  - Limitan por completo mi privacidad
- El Impuesto Microsoft
  - No puedo elegir una computadora de marca sin que incluya Windows. ¿Por qué?
- 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 sistemas 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)

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?

El control de calidad en proyectos de Software Libre

1. ¿Qué es Software Libre?
2. Comprendiendo la crisis en la industria del software
3. Características del desarrollo en proyectos de Software Libre
4. Herramientas de desarrollo colaborativo distribuído
5. Un ejemplo: coordinación y control de calidad en el proyecto Debian

Cada comunidad de desarrolladores ha llevado la filosofía de desarrollo del sistema en que creció a su modo general de trabajo y colaboración

Software Libre

- Muchos componentes pequeños, con interfaces claras, fáciles de "interenchufar", que saben hacer muy pocas cosas pero la hacen bien
- Cada componente debe mantenerse independiente de los demás. Los cambios en un componente no deben afectar a otros.
- La funcionalidad puede ser redundante - puede haber más de una herramienta para hacer lo mismo
- Asume que el usuario es como el programador - Todo está documentado y al alcance
- Al permitir al usuario hacer cosas aparentemente tontas le permitimos hacer cosas muy ingeniosas. El usuario sabe lo que hace.

Software privativo

- Grandes sistemas, con mucha funcionalidad, a ser utilizados de la manera que fue previsto. Interoperabilidad limitada a lo determinado adecuado
- El sistema está fuertemente integrado para ofrecer una sensación de mayor unidad al usuario
- El sistema es diseñado como un todo, hay que evitar al usuario tomar decisiones innecesarias
- Asume usuarios finales, da suficiente documentación para resolver problemas simples
- Es necesario proteger al usuario de las acciones que pueda realizar que puedan hacer daño a su sistema.

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

Diferentes tipos de proyecto

Hay dos tipos principales de proyectos de Software libre - y uno no puede ser exitoso sin el otro

- Desarrollo de software
  - Desarrolla cada uno de los componentes
  - El proyecto puede ser muy simple/pequeño o muy grande (hasta cientos de desarrolladores en algunos casos, tiende a las decenas)
- Integración de software (distribuciones)
  - Toma una colección de componentes y asegura su correcto funcionamiento
  - Rara vez será de menos de una decena de desarrolladores, es típico tener varios cientos
  - Normalmente son proyectos mixtos - incluyen importantes componentes de desarrollo de software

Claro está, todo proyecto de un tipo tiene importantes componentes del otro - pero hablamos del foco principal de desarrollo.

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

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

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

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
- 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 (1)

El Software Libre también presenta ciertas dificultades y retos que superar

- Coordinación de proyectos geográficamente dispersos
  - Zonas horarias
  - Los desarrolladores pueden no tener un idioma común
    - La importancia de 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
- ¿Cómo se coordina un proyecto?
  - Proyectos simples, soluciones simples
  - Proyectos complejos...
    - Líder de proyecto
    - Responsables por subconjuntos de funcionalidad
    - Responsables de rama / RMs / Pumpking / ...
    - Core team (equipos núcleo) + desarrolladores periféricos

3. Características del desarrollo en proyectos de Software Libre

Problemáticas inherentes al Software Libre (2)

- 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
- ¿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

El control de calidad en proyectos de Software Libre

1. ¿Qué es Software Libre?
2. Comprendiendo la crisis en la industria del software
3. Características del desarrollo en proyectos de Software Libre
4. Herramientas de desarrollo colaborativo distribuído
5. Un ejemplo: coordinación y control de calidad en el proyecto Debian

Depósito de código

Un proyecto colaborativo, especialmente si es geográficamente disperso, 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

- Herramientas principales
  - CVS - Concurrent Versioning System
  - SVN - Subversion
  - Arch
  - Git
  - ...

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
- Deseable: 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
  - ...

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

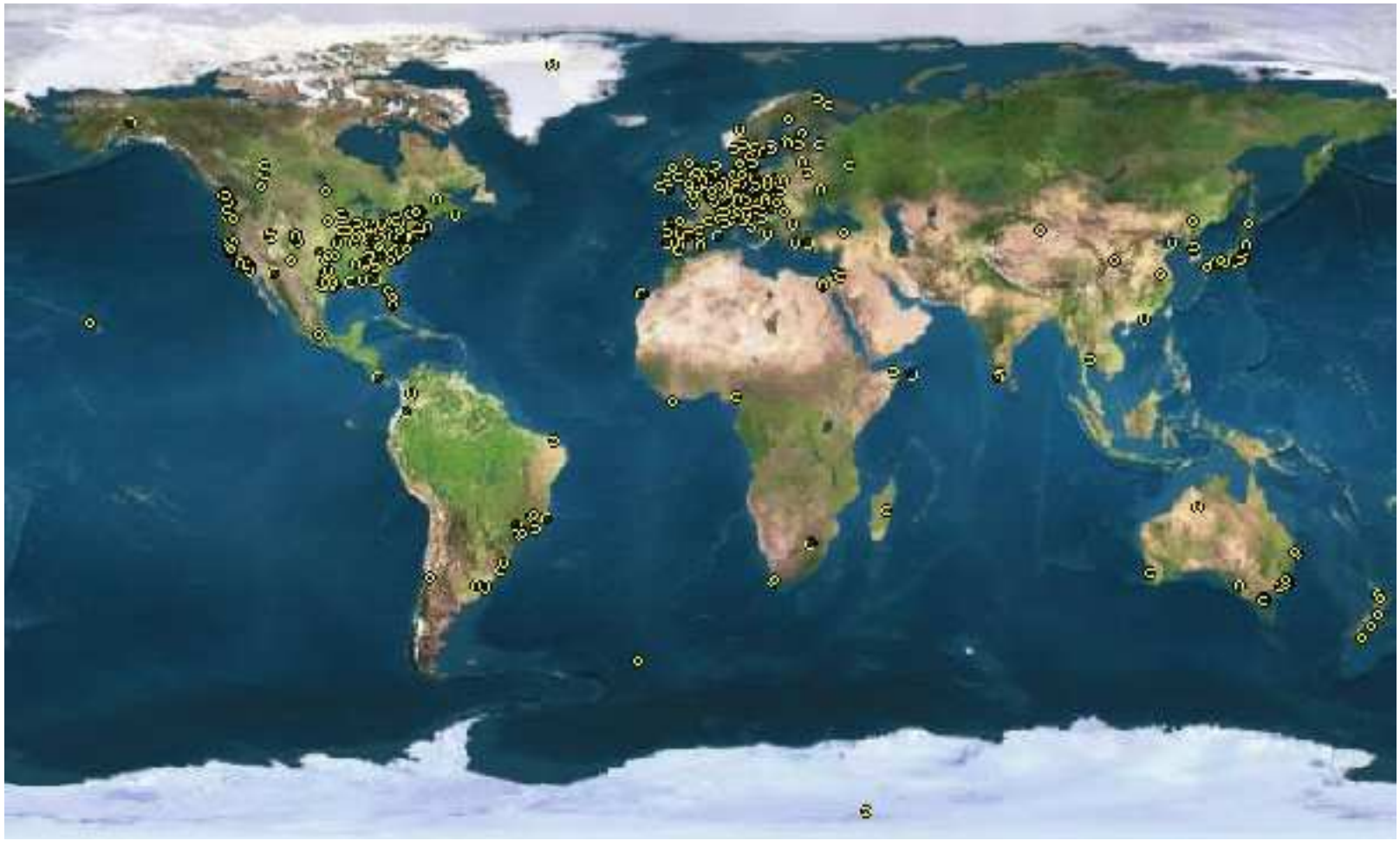
- Conversaciones uno a uno o muchos a muchos
- Capacidad de archivar fácilmente las conversaciones
- Facilidad de jugar con el sistema (bots, scripts etc.)
- Redes públicas que no requieran registro para emplearlas

El control de calidad en proyectos de Software Libre

1. ¿Qué es Software Libre?
2. Comprendiendo la crisis en la industria del software
3. Características del desarrollo en proyectos de Software Libre
4. Herramientas de desarrollo colaborativo distribuído
5. Un ejemplo: coordinación y control de calidad en el proyecto Debian

¿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
- Más de 900 desarrolladores oficiales en todo el mundo



¿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
  - Constitución
    - Cómo está estructurado el proyecto
    - Cómo podemos tomar una resolución conjunta
- Grupos de trabajo
  - Seguridad
  - Ftp-master
  - Keyring
  - QA
  - Debian-admin
  - ...

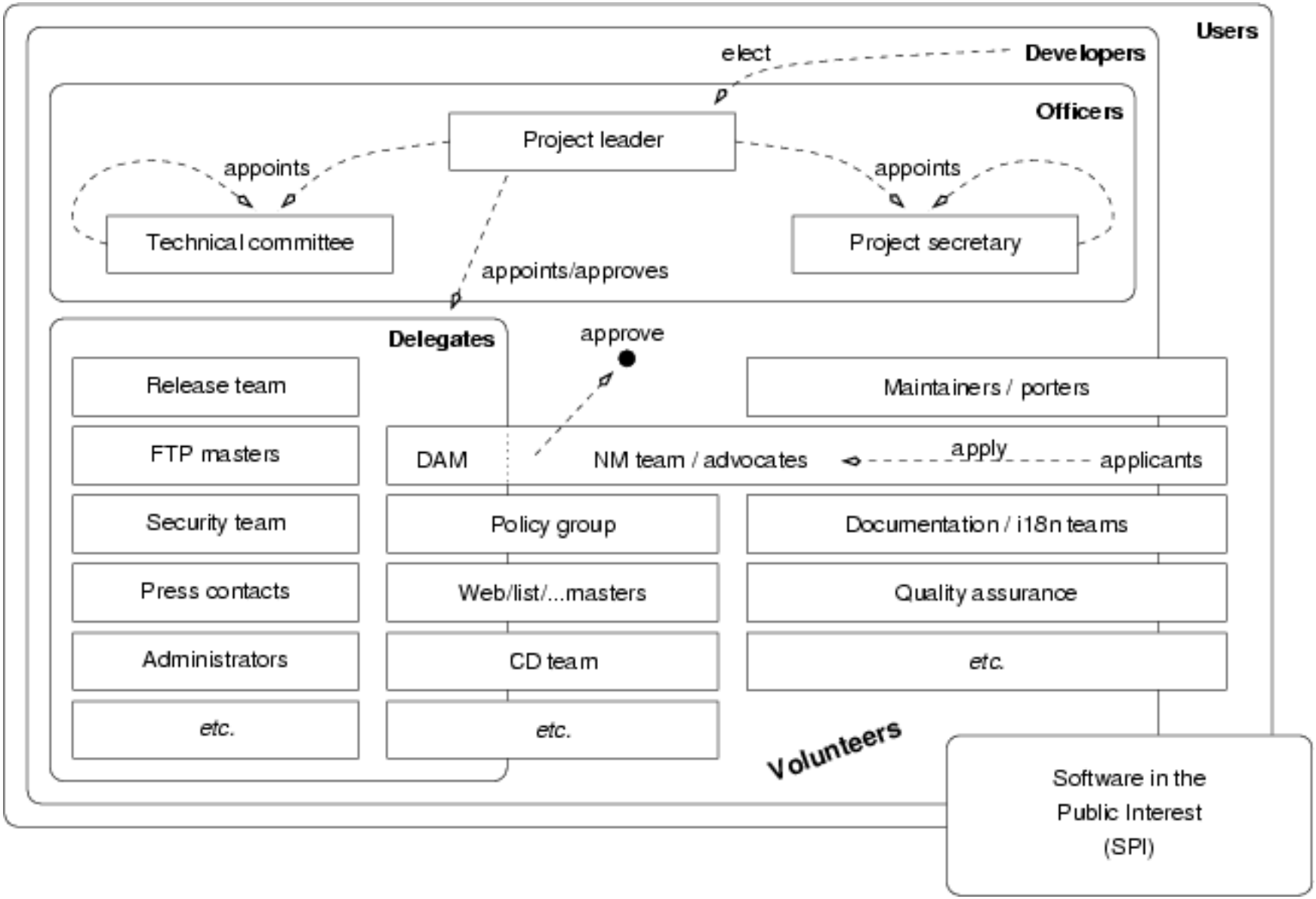
En Debian pueden colaborar tanto miembros oficiales (DDs) como externos.

- ¿Quién es un DD?
  - Quien tiene su llave PGP/GPG incluída en el llavero del proyecto
  - Quien ha pasado por los pasos de NM (Nuevo Mantenedor)
    - Identificación
    - Asignación
    - Filosofía y procedimientos
    - Tareas y habilidades
    - Aprobación del AM
    - Aprobación del DAM
- Un DD tiene pocas prerrogativas que un externo no
  - Subir directamente modificaciones al archivo
  - Dirección de correo @debian.org
  - Derecho a voto en resoluciones generales, elección del líder de proyecto
- Números
  - Actualmente hay unos 900 DDs
  - Unos 400 no-DDs que mantienen paquetes en Debian
  - Una gran comunidad alrededor

5. Un ejemplo: coordinación y control de calidad en el proyecto Debian

Organización de Debian

El proyecto tiene una jerarquía organizacional formal, aunque bastante plana:



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
  - Cualquier modificación a la rama estable debe ser aprobada por el encargado de la versión estable y por el equipo de seguridad
  - Conforme haga falta, se hacen versiones estables menores (p.ej., 3.0r1, 3.0r2 a 3.0)
  - La versión provista de cada paquete no cambia, las correcciones en versiones nuevas son aplicadas a versiones anteriores
  - Cualquier fallo de seguridad es prioritario para su inclusión en estable
- 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
  - Se recomienda para quien quiera colaborar con el desarrollo como usuario
  - En todo momento debe estar cerca de ser congelable para convertirse en la próxima versión estable
  - Muy baja prioridad para actualizaciones de seguridad - No usar en servidores

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

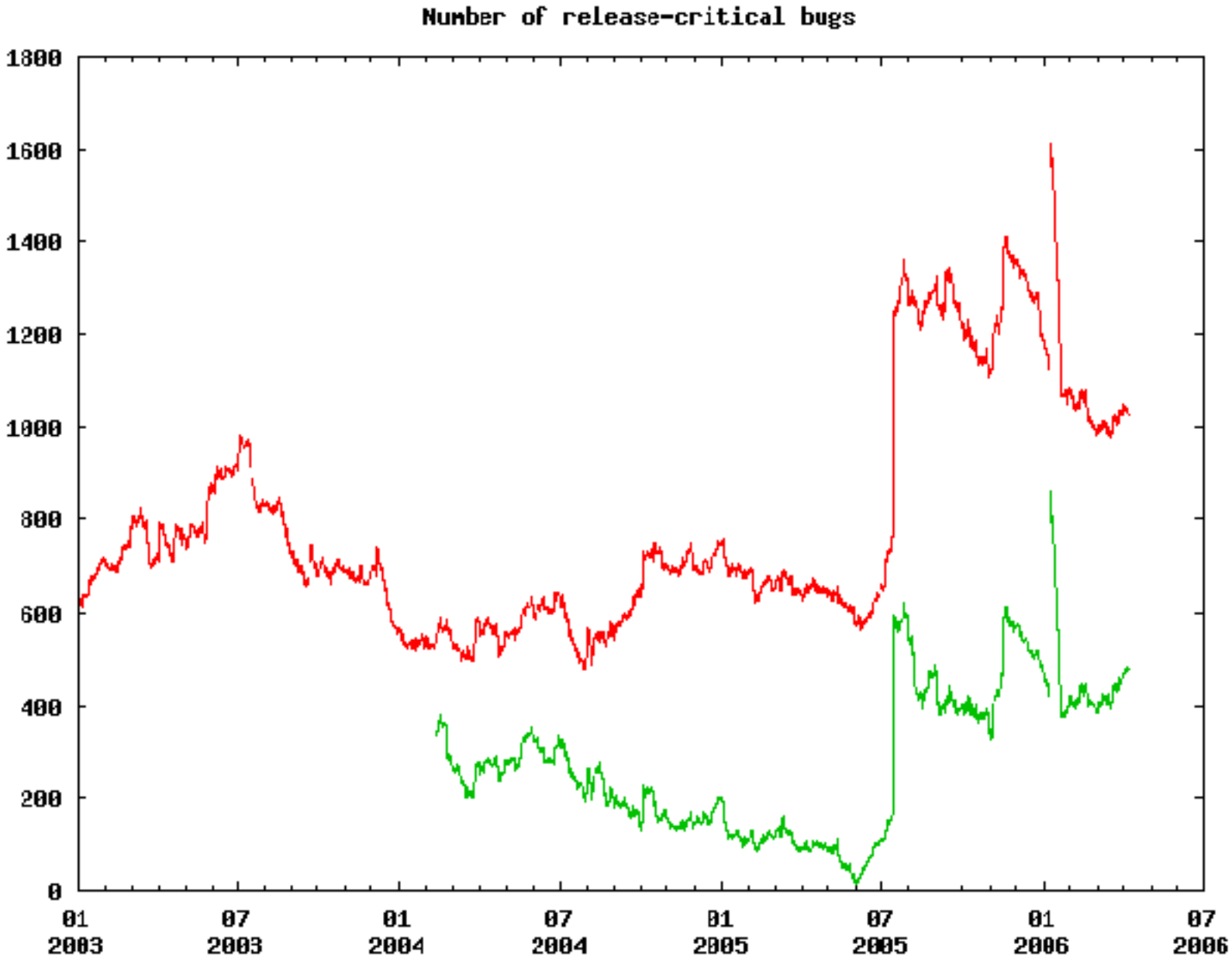
- volatile.debian.net
  - Para software cuya naturaleza hace impráctico el manejo de versiones estables con un ciclo de vida largo (p.ej. herramientas de monitoreo de seguridad, antivirus, antispam, etc.)
- backports.org
  - Versiones actualizadas de paquetes existentes en la rama estable, mantenidas 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 caso AMD64
  - La versión estable actual de Debian fue liberada sin soporte oficial para la arquitectura AMD64, por el largo periodo de *congelamiento* del sistema base
  - Un grupo importante de desarrolladores oficiales mantienen esta distribución de manera *paraoficial*
  - Hoy en día, Debian asume el trabajo del equipo AMD64 como una arquitectura con soporte oficial

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

5. Un ejemplo: coordinación y control de calidad en el proyecto Debian

BTS - Número de bugs críticos contra la rama inestable (y de pruebas) a junio 2005



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
- 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.)

¿Cómo aseguramos un nivel óptimo de calidad?

- Cada desarrollador es responsable de los paquetes que ha introducido o adoptado
  - Mantener el paquete al día
  - Corregir los fallos que pueda presentar
  - Manejar comunicación con el usuario
  - Manejar comunicación con los desarrolladores del programa en cuestión ("upstream developer")
  - Tarea de la manutención grupal para ciertos paquetes
  - ...Pero un desarrollador puede declarar huérfano a algún paquete, o desaparecer del proyecto
    - Tenemos la responsabilidad de avisar cuando estamos de vacaciones o inalcanzables

- Cada desarrollador puede corregir fallos en paquetes que no son suyos
  - Todo desarrollador puede subir un NMU (Non-Maintainer Upload) de cualquier paquete, siempre y cuando se haga responsable de sus modificaciones
  - Un NMU se reconoce por un número de versión diferente (w.x-y.z en vez de w.x-y)
  - Un NMU no marca un fallo en el BTS como ’cerrado’ sino que como ’corregido’
  - El desarrollador que corrige un fallo en NMU manda el *parche* que aplicó al BTS
  - Si un paquete muestra falta de mantenimiento, puede ser secuestrado (previa consulta en debian-devel)
  - Tradicionalmente, los NMUs son enviados con un diferimiento de 7 días, aunque estamos evaluando una política de instalación inmediata

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

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
  - ¿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.
  - Estamos trabajando en mecanismos para volver a reducirlo, probablemente a 18 meses
- 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?
  - Nuestra supuesta mayor debilidad podría convertirse en una gran fortaleza

¿Dudas?

gwolf@gwolf.org
http://www.gwolf.org/soft/qa_soft_libre