Aseguramiento de calidad en software libre

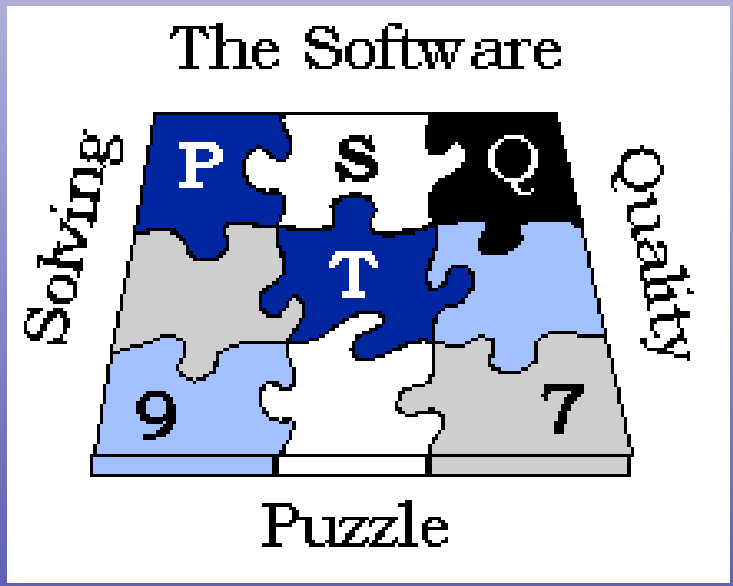
Rudy Godoy

Asociación Peruana de Software Libre

XIV CONEIS, Arequipa- Perú ago. 2005

Agenda

Calidad



Principios de desarrollo de software

Para un computador no existe concepto de código bien escrito. Sin embargo, para nosotros, humanos, es importante puesto que puede tener consecuencias en el futuro.

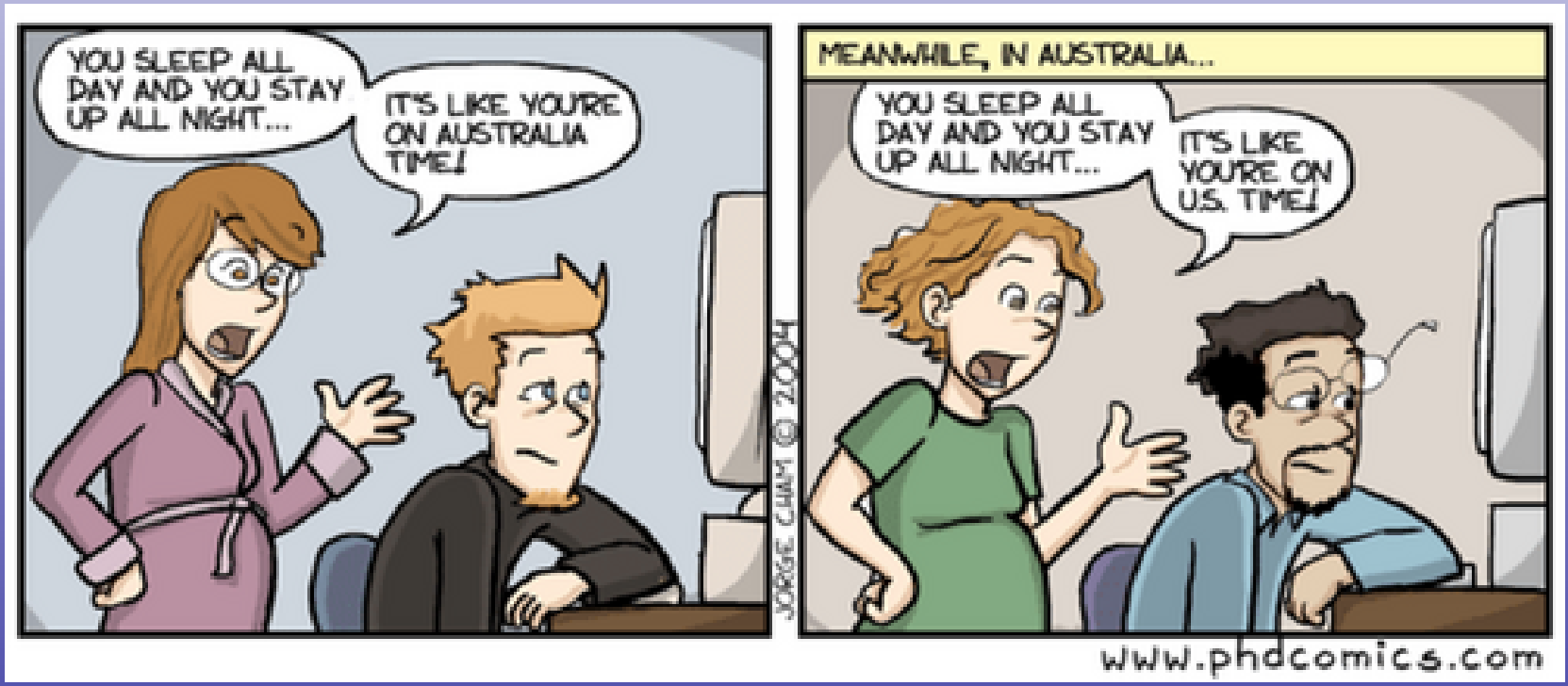
¿Qué buscamos?

- Facilidad de lectura.
- Facilidad de mantenimiento, prueba, depuración, corrección, modificación y adaptabilidad.
- Baja complejidad.
- Bajo consumo de recursos.
- Número reducido de alertas de compilación.

¿Estándares de calidad?

- **ISO 15504**
  - No define **ningún** estándar.
  - Se concentra en el modelo de capacidad de gestión y definición de procesos de las organizaciones.
  - Poca adopción debido a costo elevado y contenido no está disponible para descarga.
- **CMM(I)**
  - Documentos están disponibles para descarga en forma gratuita.
  - CMM es promovido por el Dpto. de Defensa de EEUU y por esto tiene buena aceptación en la empresas.
  - CMMI ha reemplazado a CMM e incorpora muchas de las ideas de ISO 15504, pero mantiene los beneficios de CMM.
  - Herramientas y recomendaciones para asegurar calidad en proceso de desarrollo

Desarrollo de software libre



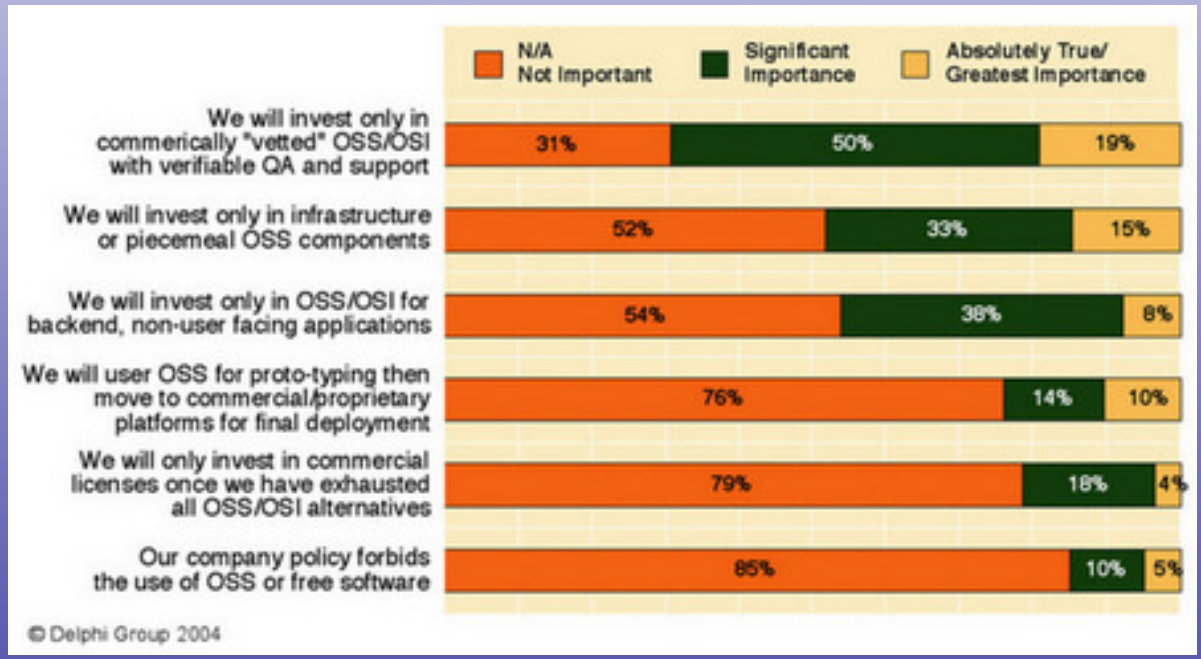
Modelo

- Iniciativa personal o de empresa: *Si algo te pica, ¡rascalo!.*
- Versiones iniciales en condición "beta": *Dejemos que la gente pruebe el software.*
- Constante evolución: *Publica con anticipación y frecuentemente.*
- Aceptación o fracaso es relativo: *Selección natural.*
- Los programas están creados bajo la filosofía Unix: *Interfaces claras e independientes del resto del sistema.*
- Se entrega toda la documentación técnica al usuario: *que es un potencial colaborador*
- Se permite al usuario realizar las cosas como las desee o necesite. *Cambiar interfaz a su gusto.*

Estado de proyectos

- Diversidad de aplicaciones de propósito similar: *Todos tienen cabida.*
- Proyectos de una sola persona: *No atrajo colaboradores o por su personalidad.*
- Cientos de aplicaciones abandonadas: *Voluntariado.*
- Diversidad de enfoques técnicos o de programación: *Mi lenguaje favorito es mejor, utilicemos ese.*
- Nivel de conocimientos de desarrolladores es muy diverso: *Programas creados por personas que recién aprendían a programar.*
- Problemas de licenciamiento: *Xfree86, qmail, mplayer*

Calidad y software libre



Mitos

- Software creado por voluntarios (léase aficionados), debe ser malo.
- No se efectúa planeamiento de necesidades de los usuarios.
- No se utiliza nigún modelo de desarrollo.
- ¿Quién me asegura que esto funciona como debe?.
- Los usuarios siempre quieren lo último.
- Los usuarios no son escuchados y no saben de que hablan.

Realidades

- Algunos de los voluntarios son: Richard Stallman, Guido Van Rossum, Alan Cox, Miguel de Icaza, Keith Packard. *Tarea: a) ¿Quién ha recibido un correo del desarrollador de su programa favorito? b) ¿Quién ha escrito MS Word?*
- Proyectos grandes reciben constante retroalimentación de los usuarios (bugzilla, debbugs, gnats).
- Los modelos de desarrollo, por ej. RUP, puede y es aplicado. XP es lo más cercano.
- Ningún tipo de software ofrece garantías (ver licencias).
- ¿Cuantas personas conocen que todavía utilizan Win9x, Win2k? (*Cabinas*)
- Todos los proyectos tienen por lo menos un medio para comunicación.

Situación actual

- Proyectos principales están realizando esfuerzos importantes de QA.
- Proyectos pequeños tienen el apoyo de los integradores.
- Gran mayoría carece de documentación sobre QA.
- Creciente preocupación por mejora en calidad y desempeño (usuario).
- Coordinación de proyectos geográficamente dispersos.
- Poco interés de comunidades locales en este aspecto.

Fortalezas y debilidades

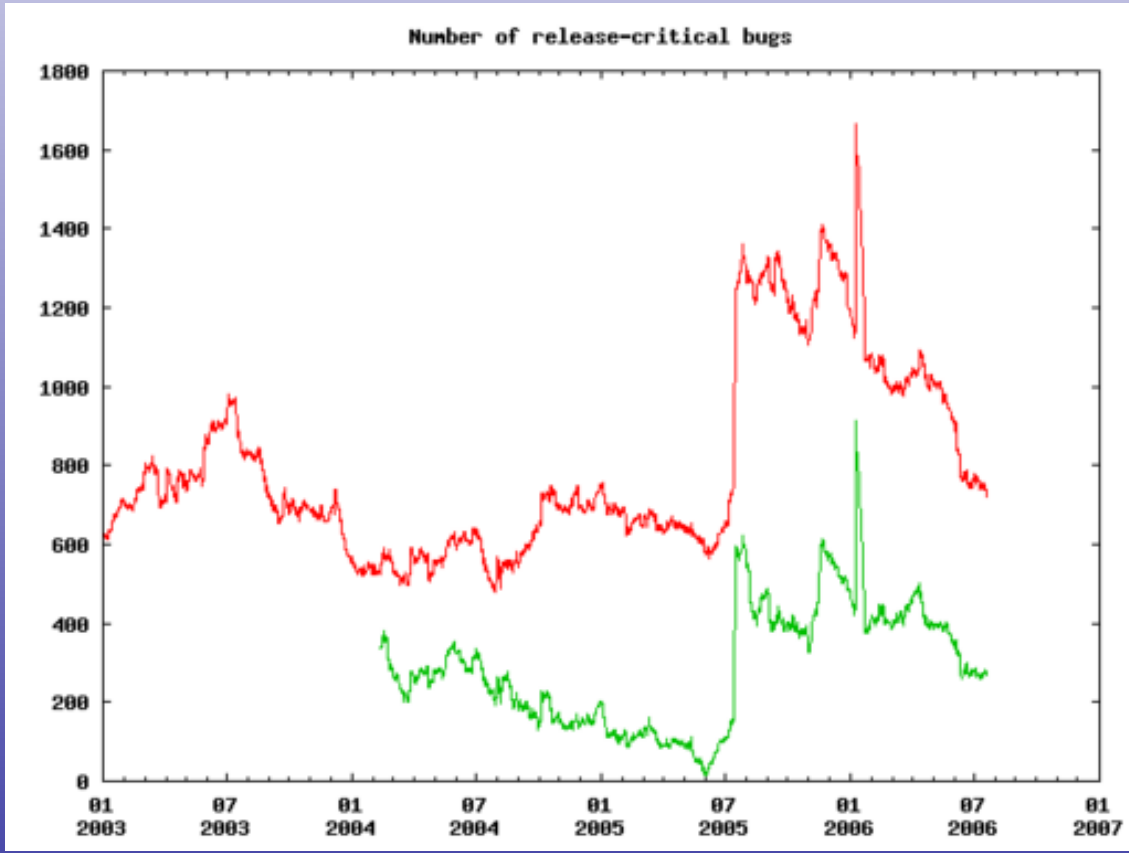
Fortalezas:

- Revisión de pares: Apertura de código y colaboración.
- Documentación de procesos (estándares) y programas (APIs).
- Optimización y mejora constante.
- Diversidad de opciones.

Debilidades:

- Proyectos más grandes o interesantes atraen mayor cantidad de personas.
- Proyectos pequeños o de pocas personas no siempre documentan procesos.
- Rápida evolución a veces duplica o divide esfuerzos.
- Desarrolladores no utilizan entornos o programas que usuarios.
- Integración es a veces dificil por decisiones de cada proyecto.

Un caso práctico: Debian



Debian y la calidad

El proceso de control de calidad en Debian empieza desde el momento que el software es empaquetado, pero no termina allí, es un proceso constante.

El control de calidad es la base fundamental del desarrollo de Debian. No hacerlo nos lleva a tener incompatibilidades y corromper el sistema.

Alcance del proyecto

- Desarrollo de un sistema operativo
- Más de 15,000 paquetes de software binarios
- Cerca de 11 arquitecturas de hardware soportadas
- Más de 900 desarrolladores en todo el mundo
- Basado en estándares, normas y constitución
- Tres ramas de desarrollo

Cómo se gestiona la calidad

- Un responsable (al 100%) por cada aplicación de software.
- Sistema de gestión de fallos completamente público
  - 275,000 fallos registrados desde 1997. Abiertos cerca de 26,000.
  - Clasificación de fallos (paquete, encargado, severidad).
- Equipo permanente de QA: BSP, MIA, WNPP, autobuild.
- Interacción y cooperación con desarrolladores de aplicaciones.
- Reuniones de trabajo, principal: DebConf, otras: Odenburg, d-i, debian-es.
- Documentación de procesos: Empaquetado, NM, QA, buildd, etc.
- Constante prueba de funcionalidad: ftpmasters: buildd, dak, desarrolladores: piuparts.

Presentación hecha con LaTeX beamer en un sistema Debian

Rudy Godoy *rudy@apesol.org.pe*