



Control de Calidad en Software Libre

III Congrés de Programari Lliure - Comunitat Valenciana



Expertos en soluciones Linux

Noviembre 2008

Juan J. Martínez

jjmartinez@opensistemas.com

- Project Manager
- Responsable de Infraestructuras en Open Sistemas
- Investigador especializado en Control Calidad y Testeo de Software



compañía española **especializada** en ofrecer

soluciones de **tecnología**

basadas en **Open Source** y plataformas **Linux**



Contenidos de la ponencia

¿Qué es 'Control de Calidad'?

Mecanismos y estrategias

Caso de estudio

Conclusiones



Control de Calidad en Software

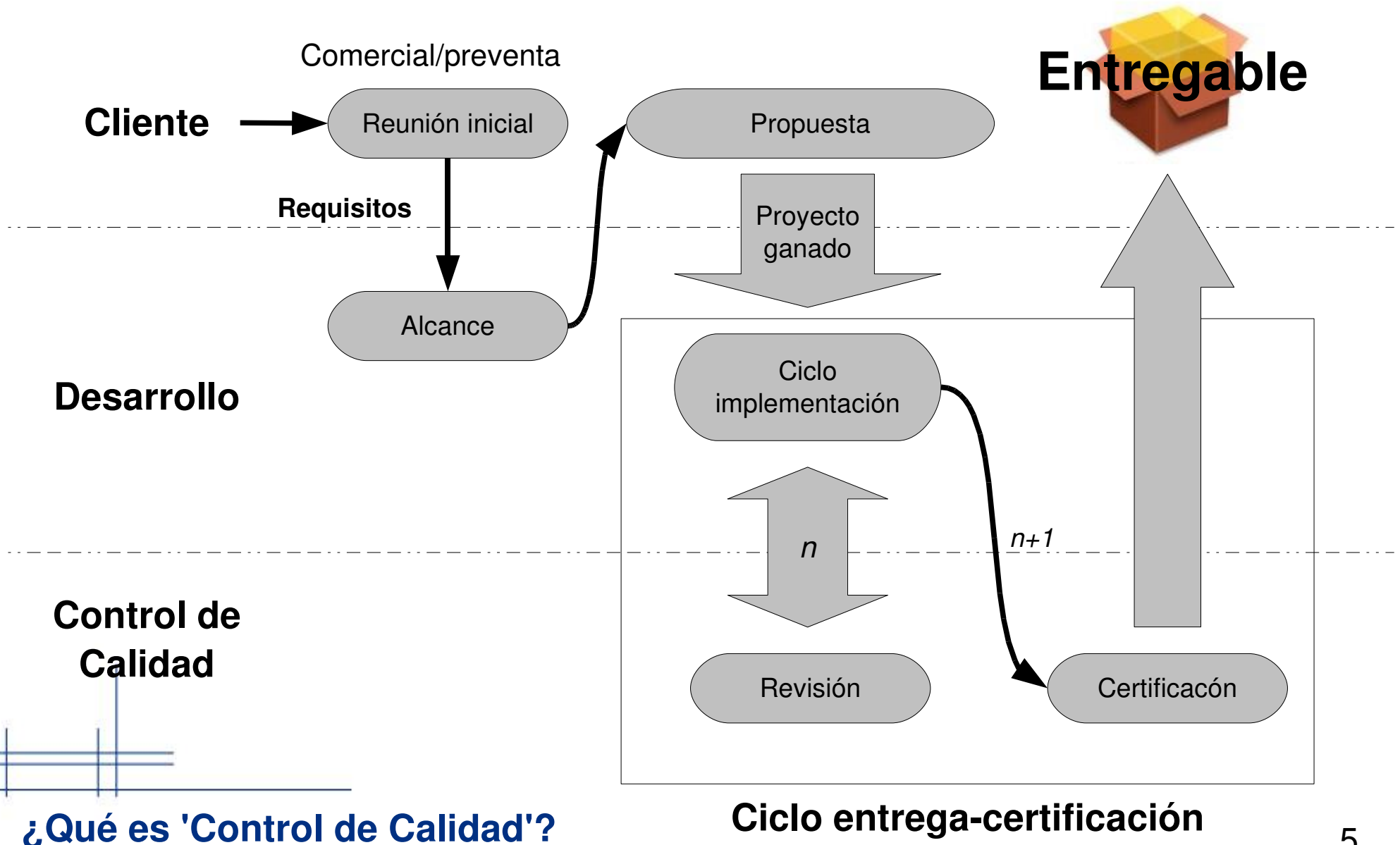
Realización de **pruebas** para detectar defectos,
bloqueando la publicación de productos defectuosos y
mejorando el resultado en **diferentes iteraciones** del ciclo
de **entrega-certificación**.

Software Quality Control



¿Qué es 'Control de Calidad'?

Control de Calidad en Software



Control de Calidad en Software

Quality Control (QC)

≠

Quality Assurance (QA)

Control de Calidad en Software

Son los mecanismos que se implican en el **proceso de desarrollo**, verificando que se sigue unos **estándares y procedimientos**, y asegurando que **los problemas se encuentran y se tratan** adecuadamente.

Software Quality Assurance

Control de Calidad en Software

"Evitar publicar con defectos"

≠

"Hacer las cosas bien a la primera"

¿Cómo encaja el Software Libre?

- Modelo Bazar (componentes de diferentes fabricantes)
- En determinados casos: componentes empaquetados por un distribuidor (actor extra)
- Software "AS IS", sin garantías

Hay Dependencias, ciclos de desarrollo desiguales en los componentes, diferentes puntos de fallo, etc.

Entonces, ¿podemos aplicar QA a nuestro producto?

Mecanismos: calidad de desarrollo

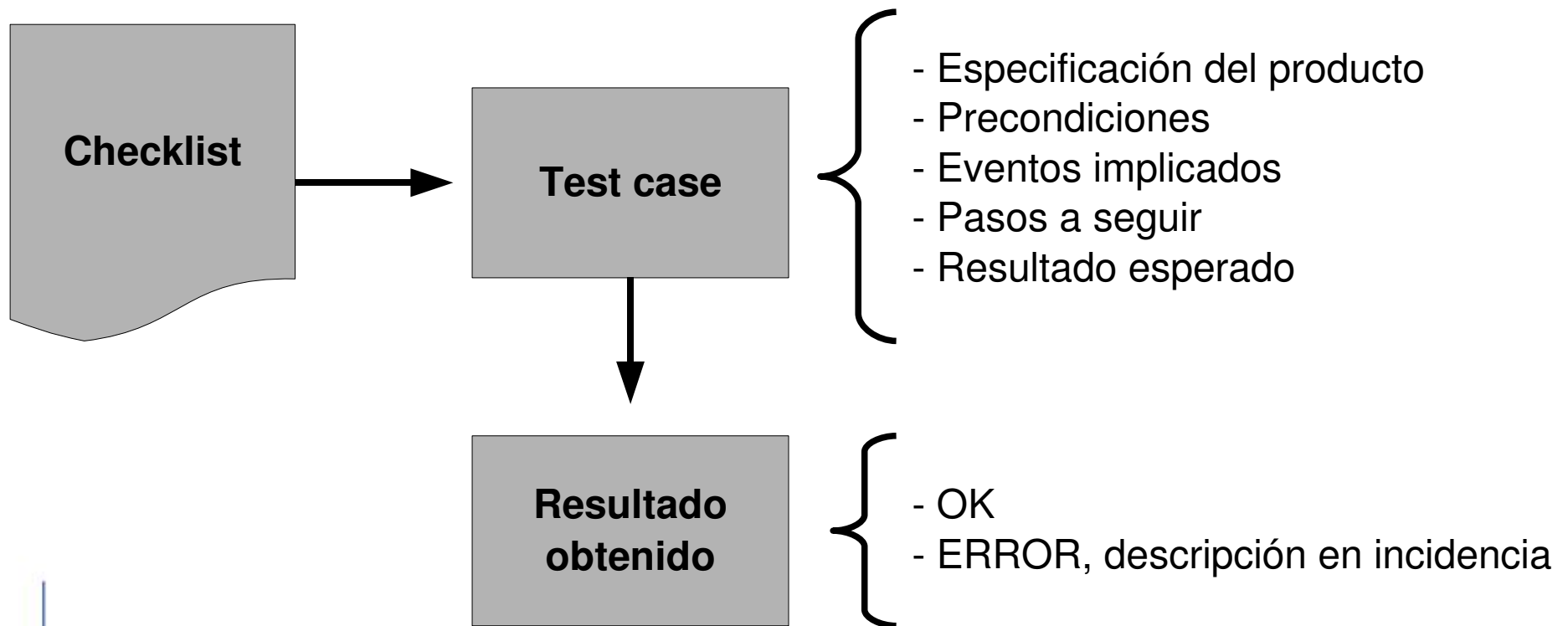
No hay grandes diferencias respecto a proyectos cerrados:

- Estilo y buenas prácticas
- Auditorías de código
- Tests unitarios
- Tests de integración
- Tests de regresión
- **Dentro del departamento de desarrollo**

Ayuda a desarrollar: detección temprana de defectos

Mecanismos: ciclos de *testing*

Tampoco hay grandes diferencias respecto a proyectos cerrados:



Mecanismos: ciclos de *testing*

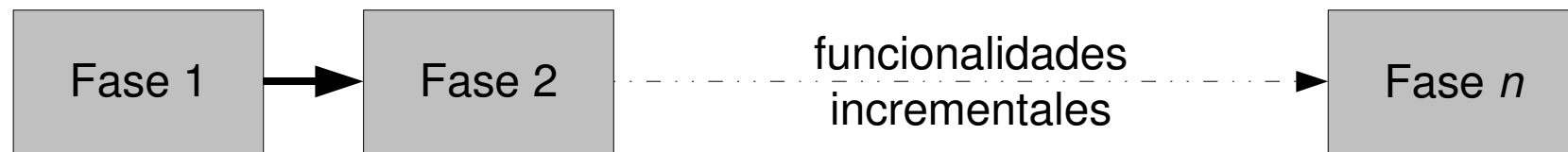
Se programan "entregas" al Departamento de Calidad previas a la versión que se entregará al cliente:

- Depende del modelo de planificación
- Cada entrega tiene un objetivo claro, y se puede repetir si no alcanza unos resultados concretos
- **Departamento independiente**, desarrollo no interfiere

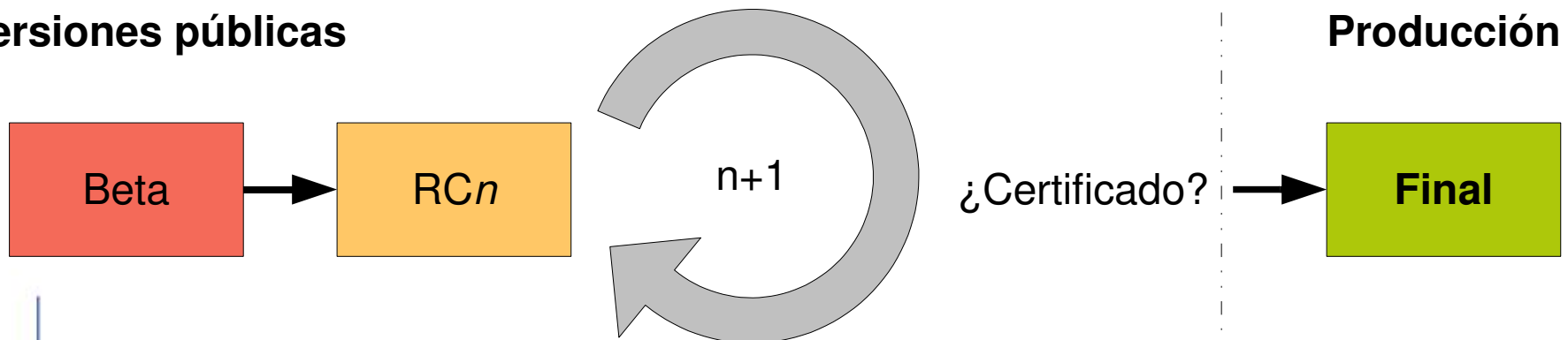
Mejora producto: se detectan errores antes de publicar

Mecanismos: ciclos de *testing*

Ejemplo de planificación de entregas al Departamento de Calidad:



Versiones públicas



Mecanismos: automatización

Gran cantidad de *frameworks* para automatizar todo tipo de pruebas:

- Más pruebas en menos tiempo
- La linea entre desarrollo y calidad se difumina
- Pruebas nocturnas (*AKA* ¿funciona el código del repositorio?)
- En general es positivo (¿quién *prueba* a los que *prueban*? y... requiere una inversión de tiempo *difícil de justificar*)

Mecanismos: *bugtracking* público

Estamos hablando de Open Source (mantra: *release early, release often*):

- Cualquier versión *no-final* debería estar abierta al público: integrar al usuario en el "ciclo de testing"
- Debe ser fácil para el usuario apuntar a un defecto
- El usuario *no es el Departamento de Calidad*: nos dice qué algo falla, pero sus informes no serán *de calidad*
- Pasar a una nueva entrega nunca debe ser traumático
- División del área de Calidad dedicada a los informes de usuario

¿De quién es este bug?

En muchos proyectos, no está claro:

- ¿Es nuestro?
- ¿Es un fallo de integración?

Y lo más importante:
¿corresponde al *upstream*?

Es imprescindible para saber
quién lo debe corregir...



©MARLIN E. RICE

Ejemplo: distribución GNU/Linux

Características:

- Se basa en otra distribución (que a su vez empaqueta otros proyectos)
- Incorpora una serie de paquetes diferenciadores
- En sus paquetes se puede incluir desarrollos propios

Ejemplo: distribución GNU/Linux

Puntos clave:

- *No reinventes la rueda*, hay que reutilizar todo lo que se pueda de la distribución base
- Equilibrio entre integración y "añadidos", la integración se debe enviar a *upstream*
- Si se corrige un *bug* que no es nuestro, es muy importante que se incluya esa corrección en *upstream*
- Sincronizar ciclos de desarrollo

Ejemplo: distribución GNU/Linux

Seguimiento de un *bug*:

- Es vital poder hacer seguimiento de los defectos, *aunque no sea competencia nuestra resolverlos*
- Ejemplo de Ubuntu con Launchpad:

CVE Bugs: 96 (*Common Vulnerabilities and Exposures*)

Bugs fixed elsewhere: 1403

Total open: 47966

Distinción: problemas de seguridad, errores *upstream* y otros errores.

Ejemplo: distribución GNU/Linux

Bug #112102: Too easy to accidentally kill dbus from Services settings and lock yourself out of services

File Edit View History Bookmarks Tools Help

https://bugs.launchpad.net/ubuntu/+source/gnome-
Google

launpad Ubuntu "gnome-system-tools" package
Overview Code Bugs Blueprints Translations Answers

Bug #112102: Reported by Peter da Silva on 2007-05-03 (Activity log)
Too easy to accidentally kill dbus from Services settings and lock yourself out of services
This report is public

Affects	Status	Importance	Assigned to	Milestone
GST	Fix Released	Unknown	gnome-bugs #440416	
Rebuntu	Invalid	Undecided		
gnome-system-tools (Ubuntu)	Fix Committed	High	Ubuntu Desktop Bugs	

Also affects project Also affects distribution Nominate for release

Binary package hint: gnome-system-tools

From the "Services settings" dialog (services-admin), I accidentally clicked the box for "System Communication bus (dbus)" instead of "Printer service (hplip)" directly above it. This happened because the dialog had stopped responding briefly while disabling "Printer Service (cupsys)" without any indication, and a slight inadvertant movement of the scrollwheel was registered before the mouse click after it had finished disabling cupsys, moving dbus under the mouse.

Duplicates of this bug
Bug #138196

bugs.launchpad.net

Sebastien Bacher wrote on 2007-05-22: (permalink)

Thanks for your bug report. This bug has been reported to the developers of the software. You can track it and make comments here: http://bugzilla.gnome.org/show_bug.cgi?id=440416

File Edit View History Bookmarks Tools Help

http://bugzilla.gnome.org/show_bug.cgi?id=440416
Bugzilla
GNOME™ New bug · Browse · Search · Reports · Help
Log In | New Account

Bug 440416 – Too easy to accidentally kill dbus from Services settings and lock yourself out of services
[View Bug Activity](#)

Opened by Sebastien Bacher (reporter, developer) (points: 27)
2007-05-22 10:00 UTC

The bug has been opened on
<https://bugs.launchpad.net/ubuntu/+source/gnome-system-tools/+bug/112102>

Binary package hint: gnome-system-tools

From the "Services settings" dialog (services-admin), I accidentally clicked the box for "System Communication bus (dbus)" instead of "Printer service (hplip)" directly above it. This happened because the dialog had stopped responding briefly while disabling "Printer Service (cupsys)" without any indication, and a slight inadvertant movement of the scrollwheel was registered before the mouse click after it had finished disabling cupsys, moving dbus under the mouse.

Immediately Services closed and attempts to re-run it produced the message "you are not allowed to access the system configuration".

Done

Control de Calidad y Software Libre

Desde nuestro desarrollo:

- Integrar estrategias de control de desarrollo (tests unitarios, de integración y de regresión)

Desde Control de Calidad:

- Ciclos entrega-certificación no demasiado largos
- Integrar al usuario en los ciclos de *testing*
- Gestionar adecuadamente los errores (comunicación y colaboración con el *upstream*)

¿Alguna pregunta?

Gracias por venir

Juan J. Martínez

jjmartinez@opensistemas.com

Empresa: <http://www.opensistemas.com/>

Blog personal: <http://blackshell.usebox.net/>