

# Gestión y calidad de Software de Fuentes Abiertas/Libre

# Integrantes:

1. Borello, Agustin
2. Donato, Alexis
3. Lopez, Marcos
4. Mattio, Paolo
5. Vilardo, Milena

# Concepto básico de un proyecto de software libre

Un proyecto de fuentes abiertas consiste en una colección de programadores al azar con diferentes mentalidades que muy probablemente nunca se han visto, y que tienen objetivos personales muy diferentes para trabajar en el proyecto.



# Punto de Debate

---

¿Consideran que es fácil llevar a cabo un proyecto de software libre?



Un proyecto de Software Libre implica:

- Acomodar el código para que sea comprensible
- Crear la documentación de desarrollo (Pautas de desarrollo) y una lista de correos
- Redactar la documentación del proyecto
- Realizar una presentación del sitio web adecuada



COMUNICACIÓN COLABORACIÓN

# Ley de Brooks

---

Fred Brooks observó que la complejidad de las comunicaciones en un proyecto se incrementa al cuadrado del número de participantes.

# Factores a tener en cuenta para la GESTIÓN

1. Organización del proyecto
2. Canales de comunicación
3. Metodología de contribución
4. Infraestructura de alojamiento del proyecto
5. Sistema de control de versiones

# Punto de Debate

---

¿Quién toma las decisiones del proyecto?

¿Quién define las estrategias a seguir?

¿Quién aprueba contribuciones?



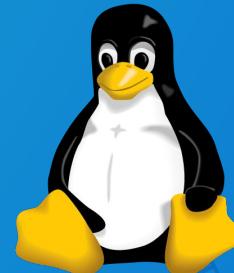
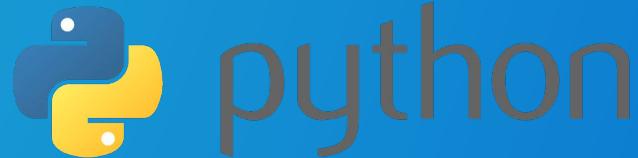
# Monarquista

Una persona lleva a cabo la mayor parte de las decisiones.

"Dictador Benevolente"



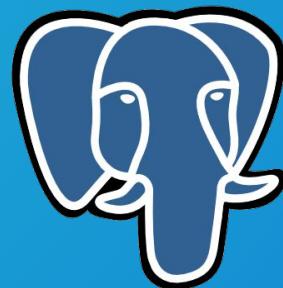
Linus Torvalds



# Comunitaria

---

El proyecto es manejado por un conjunto de colaboradores de manera democrática, tomando decisiones bajo consenso.



PostgreSQL

# Corporativa

---

El proyecto es manejado por una empresa privada.  
Generalmente surge de un producto comercial que se  
vuelve software abierto.



# Fundación

---

Fundaciones sin fines de lucro que surgen para dar soporte y liderar grandes proyectos. Algunas son meritocráticas



# Meritocracia

---

Las decisiones importantes son tomadas por aquellas personas que han contribuido de manera importante y han ganado credibilidad en su trabajo.

Meritocracia = forma de gobierno basada en el mérito



redhat<sup>®</sup>

## 2. Canales de comunicación

---

- IRC
- Mail
- Foros
- Blogs



# Punto de Debate

---

¿Sólo se puede contribuir a un proyecto de software libre agregando código o arreglando bugs?



### 3. Formas de contribución

---

- Escritura de tutoriales, documentación, blogs.
- Resolución de dudas/consultas.
- Realizar revisiones de código.
- Moderación de canales de comunicación.
- Difundir el proyecto a través de medios de comunicación.
- Realizar investigación de usuarios para mejorar usabilidad.

## 4. Infraestructura de alojamiento en la Web

Servicios de web-hosting para el proyecto, brindando herramientas que ayudan a la gestión del mismo.



**source  
forge**



## 5. Sistema de control de versiones

Gestión de los cambios que se realizan sobre elementos del proyecto.



# Control de Calidad en Software Libre

Evitar publicar con defectos (QC)

≠

Hacer las cosas bien a la primera (QA)

Software libre: modelo Bazar, componentes empaquetados por un distribuidor, software "AS IS", Dependencias, ciclos de desarrollo desiguales, diferentes puntos de fallo, etc.

# Diferencias en control de calidad entre Software libre y cerrado

No hay grandes diferencias respecto a:

- Calidad de desarrollo dentro del departamento de desarrollo: testing unitario raro que se incluya
- Ciclos de testing en un departamento independiente a desarrollo (proceso de entrega)
- Automatización: generalmente la agregan los propios usuarios

# Bugtracking público

---

Open Source (mantra: release early, release often)

En algunos proyectos grandes hay máquinas exclusivas dedicadas a compilar código más reciente

Versión no final: se integra al usuario en el “ciclo de testing”

# Bugtracking público

---

En muchos proyectos no está claro de quién es el bug  
Upstream: envío de parche o corrección a autor original o  
mantenedores principales. Errores de integración y que no  
sean nuestros

Distinción: seguridad, upstream, otros

# Bugtracking público

---

Equilibrio entre integración y "añadidos"

Sincronizar ciclos de desarrollo

# CENDITEL - Aseguramiento de Calidad en el SWL

Énfasis en la mejora de la práctica de documentación.

Subprocesos:

- Evaluación del proceso de desarrollo.
- Evaluación de la calidad en las aplicaciones.



# Evaluación del Proceso de Desarrollo

---

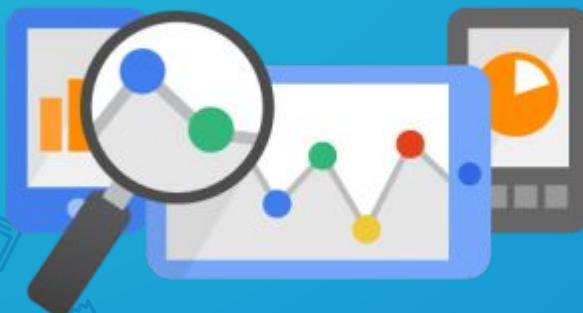
Objetivo: Mejora continua, detectando deficiencias y promoviendo el seguimiento de estándares.

**ISO/IEC  
15504**



# Evaluación de la Calidad en las Aplicaciones

Objetivo: Evaluar el cumplimiento de requerimientos de calidad, establecido en función de métricas.



*Se mide en término de RF y RNF.*

# Generación de Reportes de Aseguramiento de Calidad en el Desarrollo de SWL

Objetivo: Informar a los equipos las aplicaciones evaluadas y discrepancias observadas.



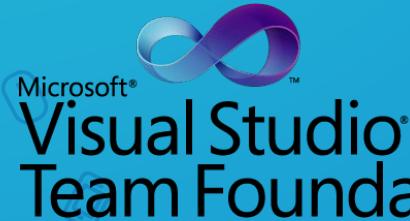
# Continuous Integration



- ¿Qué es?
- Surgimiento de la idea: Grady Booch.
- Primer sistema CI desarrollado

# Continuous Integration

Algunos ejemplos de sistemas CI



**Travis CI**



**Jenkins**

# Continuous Integration

---

Análisis de su impacto en proyectos



- Hilton, Michael
- Tunnell, Timothy
- Huang, Kai
- Darko, Marinov
- Danny, Dig

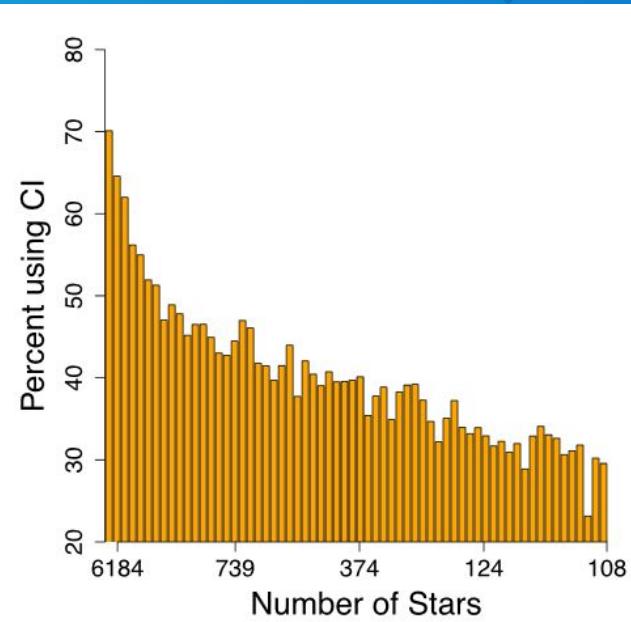
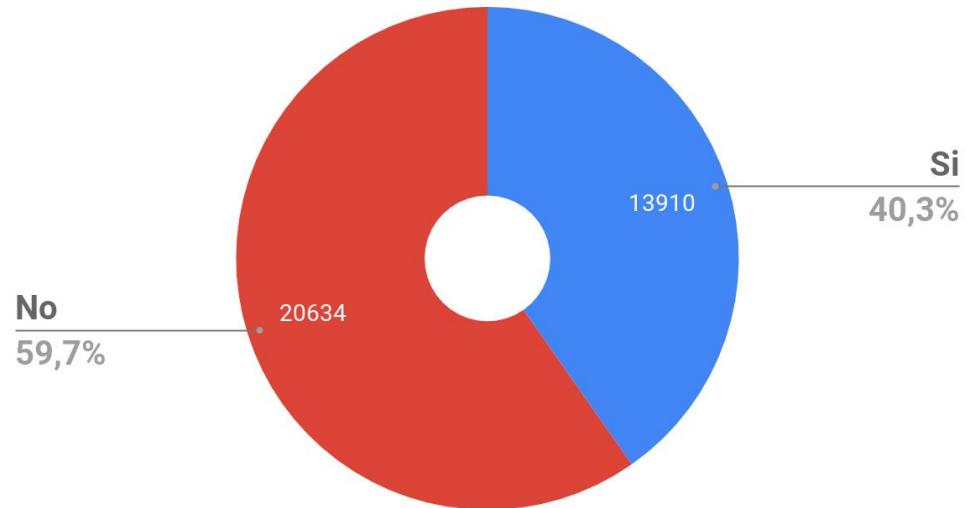
# Punto de Debate

- ¿En qué porcentaje creen que los proyectos utilizan sistemas de Integración continua?
- ¿La popularidad de un proyecto afecta en la decisión de utilizar sistemas de integración continua?
- ¿Y los lenguajes empleados?
- ¿Por qué razón implementarían CI?
- ¿Por qué razón no lo harían?



# Continuous Integration

Porcentaje de Utilizacion de Sistemas de Integracion Continua



# Continuous Integration

Language	Total Projects	# Using CI	Percent CI
Scala	329	221	67.17
Ruby	2721	1758	64.61
Go	1159	702	60.57
PHP	1806	982	54.37
CoffeeScript	343	176	51.31
Clojure	323	152	47.06
Python	3113	1438	46.19
Emacs Lisp	150	67	44.67
JavaScript	8495	3692	43.46
Other	1710	714	41.75
C++	1233	483	39.17
Swift	723	273	37.76
Java	3371	1188	35.24
C	1321	440	33.31
C#	652	188	28.83
Perl	140	38	27.14
Shell	709	185	26.09
HTML	948	241	25.42
CSS	937	194	20.70
Objective-C	2745	561	20.44
VimL	314	59	18.79

Table 4: Reasons developers gave for not using CI

Reason	Percent
The developers on my project are not familiar enough with CI	47.00
Our project doesn't have automated tests	44.12
Our project doesn't commit often enough for CI to be worth it	35.29
Our project doesn't currently use CI, but we would like to in the future	26.47
CI systems have too high maintenance costs (e.g., time, effort, etc.)	20.59
CI takes too long to set up	17.65
CI doesn't bring value because our project already does enough testing	5.88

Table 6: Reasons for using CI, as reported by survey participants

Reason	Percent
CI makes us less worried about breaking our builds	87.71
CI helps us catch bugs earlier	79.61
CI allows running our tests in the cloud, freeing up our personal machines	54.55
CI helps us deploy more often	53.32
CI makes integration easier	53.07
CI runs our tests in a real-world staging environment	46.00
CI lets us spend less time debugging	33.66

# Herramientas para la Automatización

---



Jenkins



Created by  
ThoughtWorks®



# Sistemas de seguimiento de Bugs

---



# Bugzilla



# Bibliografía

---

- Introducción al software libre  
<http://www.cyd.conacyt.gob.mx/222/Articulos/Tecnologiasparacompartir/sobre-all.pdf>
- Fundación CENDITEL, "Aseguramiento de la Calidad en el Desarrollo de Software Libre"  
[http://calidad-sl.cenditel.gob.ve/files/2011/06/resumenProyecto\\_04092013.pdf](http://calidad-sl.cenditel.gob.ve/files/2011/06/resumenProyecto_04092013.pdf)
- Catedral y el Bazar  
<http://softlibre.unizar.es/manuales/softwarelibre/catedralbazar.pdf>
- Fogel, Karl. "Producir software de código abierto: Cómo llevar a buen puerto un proyecto de código libre"  
[https://github.com/RHMedel/inqSWL2018/blob/master/bibliografia/Fogel-Producing\\_Open\\_Source\\_Software-2007.pdf](https://github.com/RHMedel/inqSWL2018/blob/master/bibliografia/Fogel-Producing_Open_Source_Software-2007.pdf)
- Martínez, Juan J., "Control de Calidad en Software Libre"  
[https://blackshell.usebox.net/pub/misc/ControlCalidadSL\\_Lliurex.pdf](https://blackshell.usebox.net/pub/misc/ControlCalidadSL_Lliurex.pdf)
- González Barahona, Jesús, Seoane Pascual, Joaquín, Robles, Gregorio. "Introducción al software libre", Fundació per a la Universitat Oberta de Catalunya, 2003. ISBN: 84-9788-028-5  
[http://www.eoi.es/wiki/index.php/Procesos\\_en\\_el\\_software\\_libre\\_en\\_Software\\_libre](http://www.eoi.es/wiki/index.php/Procesos_en_el_software_libre_en_Software_libre)
- The 5 Types of Open Source Projects  
<https://wackowiki.org/doc/Org/Articles/5TypesOpenSourceProjects>
- How to Contribute to Open Source  
<https://opensource.guide/how-to-contribute/>
- Usage, Costs, and Benefits of Continuous Integration in Open-Source Projects  
Hilton Michael, Tunnell Timothy, Huang Kai, Darko Marinov y Danny Dig (2016)

# Viva el software libre!... na mentira

