

---

# GNOME como caso de estudio de Ingeniería del Software Libre

*Juan José Amor Iglesias, Gregorio Robles  
Martínez, Jesús M. González-Barahona*

*{jjamor,grex,jgb} \_at\_ gsync.escet.urjc.es*



*Mayo 2005*

---

(cc) 2005 J. J. Amor, G. Robles, J. González-Barahona.  
Some rights reserved. This work licensed under Creative Commons  
Attribution-ShareAlike License. To view a copy of full license, see  
<http://creativecommons.org/licenses/by-sa/2.0/>

# Índice

- Ingeniería del Software Libre. CALIBRE.
- Proceso de Desarrollo.
- GNOME: Código fuente.
- Lenguajes en GNOME.
- Desarrolladores en GNOME. Desigualdad. Generaciones.
- Contribución de cada desarrollador.
- Varias fuentes de datos simultáneas.
- ¡Demostración! (si la red lo permite...).
- Conclusiones.

## Software Libre: Ingeniería

Software libre:

- Cientos, miles de personas distribuidos geográficamente.
- Que apenas se ven (salvo alguna reunión como Guadec-ES).
- Coordinación. Productividad. Capacitación.
- Producen resultados que ¡funcionan!

¿Es esto mágico? ¿Cómo es el proceso de desarrollo del software libre?

¿Cómo podemos mejorarlo?

- Grupos de investigación universitarios.
- Grupo de Ingeniería del Software Libre en URJC.

## CALIBRE

- Acción coordinada europea. Sexto programa marco IST.
- Integrar y Coordinar investigación europea de Software libre en “mercado secundario”.
- Participan: 8 universidades europeas y algunos partners industriales.
- URJC: Caracterización de procesos, productos y proyectos de software libre.
- Estudios cualitativos y cuantitativos.
- <http://calibre.ie/>

## Modelo de desarrollo software libre

El modelo: distribuido, colaborativo, abierto... en Internet.

Hacen falta herramientas que faciliten ejecución del modelo: son las que proporcionan gran cantidad de datos accesibles públicamente.

Fuentes de datos utilizadas:

- Código fuente: Paquetes binarios y fuentes (tarballs).
- Control de versiones: CVS, SVN...
- Seguimiento de fallos: Bugzilla...
- Documentación: man, docbook...
- Listas de correo, foros públicos.
- Estadísticas de uso (Popcon).
- Encuestas: FLOSS.

# Estudio de Código Fuente de GNOME

Origen de los datos: los *tarballs*

Cómo contamos las líneas de código:

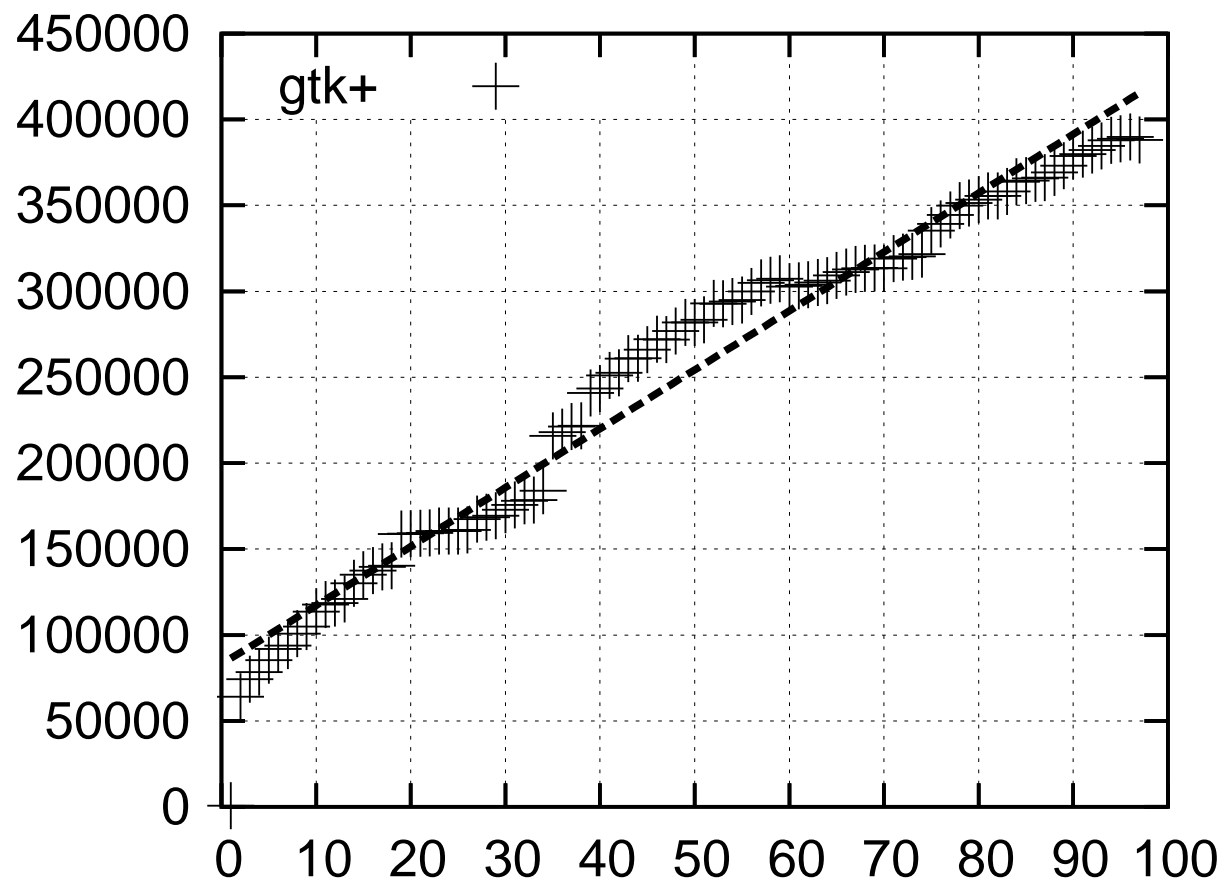
- Con `wc`
- Con scripts `awk`
- Con `SLOCCount`

La mejor opción es `SLOCCount` porque:

- Heurísticas que distinguen lenguajes.
- No cuenta documentación y otros ficheros que no sean *fuentes*.

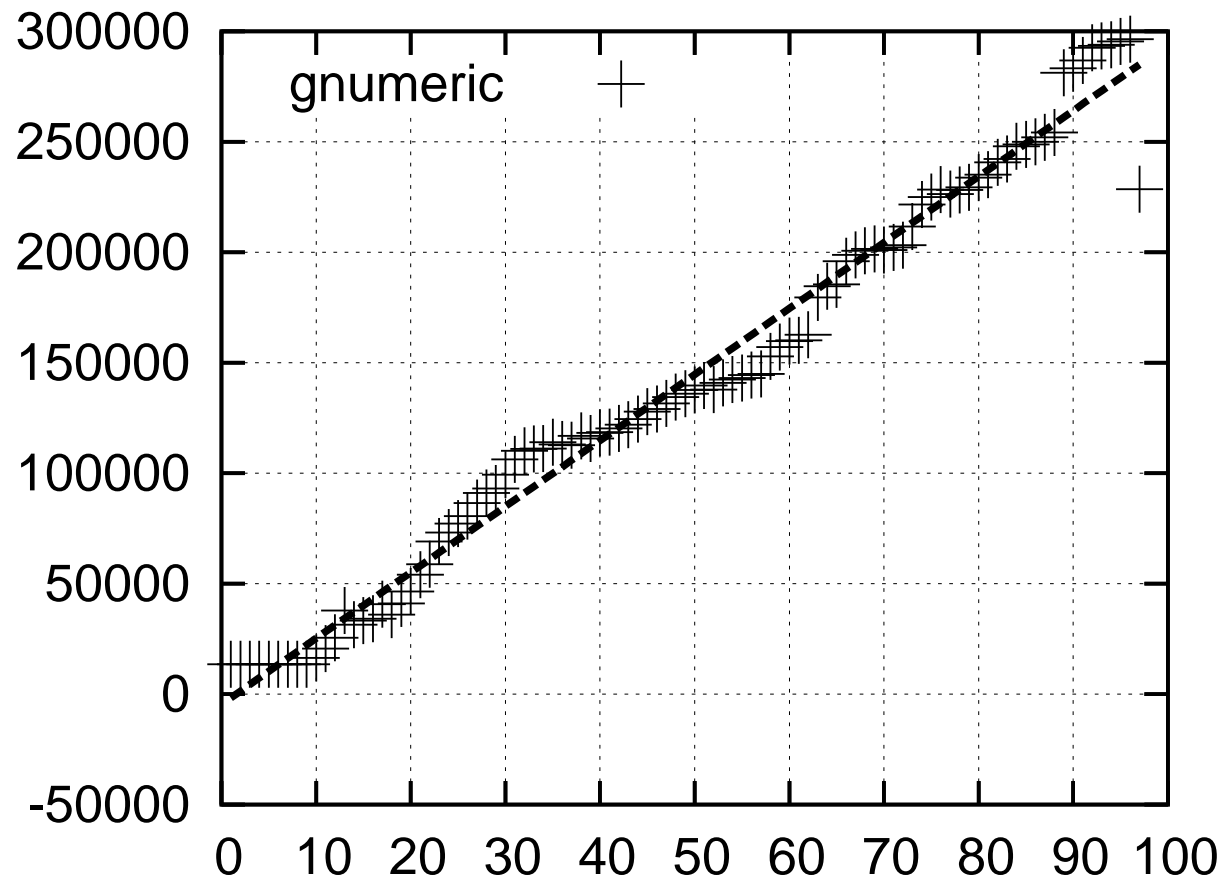
Alternativa: Contar fuentes desde CVS.

## Evolución de GTK+



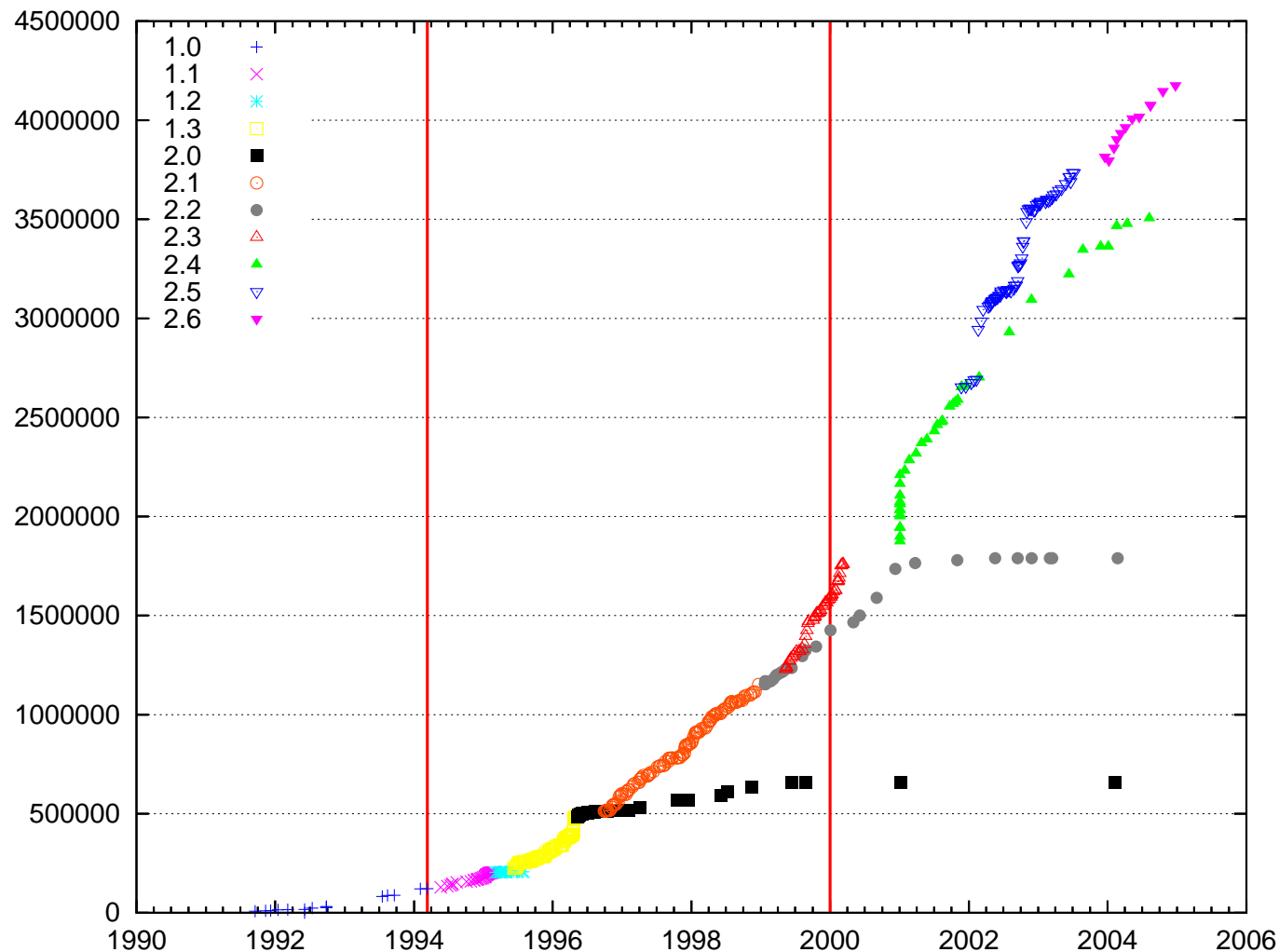
Ajuste:  $y = 3371,7 \cdot t + 89968,9; r = 0,985$

## Evolución de Gnumeric



Ajuste:  $y = 3019,9 \cdot t + 17322,8; r = 0,988$

# ¡Esto no es GNOME!



## Observaciones del código fuente

La mayoría de los proyectos de cierto tamaño como GNOME:

- Crecen linealmente.
- Rara vez crecen sub-linealmente.
- A veces super-linealmente (Linux).
  - $y = 0,26 \cdot t^2 - 322 \cdot t + 195,183; r^2 = 0,99$

¿Por qué?

- Linealidad: Organización más eficiente que en entornos clásicos.
- Superlinealidad: Crecimiento lineal de drivers + crecimiento lineal en número de drives.
- Sublinealidad: Se da poco en los grandes proyectos de software libre.

## Otros resultados desde el código fuente

Estimación de coste económico: COCOMO.

| Software  | Líneas  | Esfuerzo   | Tiempo desarrollo | Coste Total   |
|-----------|---------|------------|-------------------|---------------|
| Dia       | 122,722 | 31.23 a-h  | 1.98 años         | \$ 4,218,868  |
| Evolution | 207,507 | 54.19 a-h  | 2.44 años         | \$ 7,320,252  |
| Gnumeric  | 294,547 | 78.28 a-h  | 2.81 años         | \$ 10,574,357 |
| Gtk+      | 418,927 | 113.31 a-h | 3.23 años         | \$ 15,306,888 |

Distribución de lenguajes: la presencia de otros distintos del lenguaje “principal” es normalmente casi anecdótica.

## Estudio del CVS de GNOME

CVS:

- Sistema de control de versiones.
- Acceso remoto. Desarrollo distribuido.
- Acceso anónimo (read-only).

El acceso anónimo permite:

- Acceso a todas las versiones.
- Acceso a información de los commits (logs).
- Logs: Líneas cambiadas por fichero, fecha, hora, desarrollador.

La herramienta CVSAly analiza estos datos y permite obtener diversos resultados...

## CVS: Coeficiente de Gini

Hecho: La mayor parte del trabajo la realiza una minoría de desarrolladores.

Grado de desigualdad: (en aportaciones al proyecto por cada desarrollador). Medible con el coeficiente de Gini.

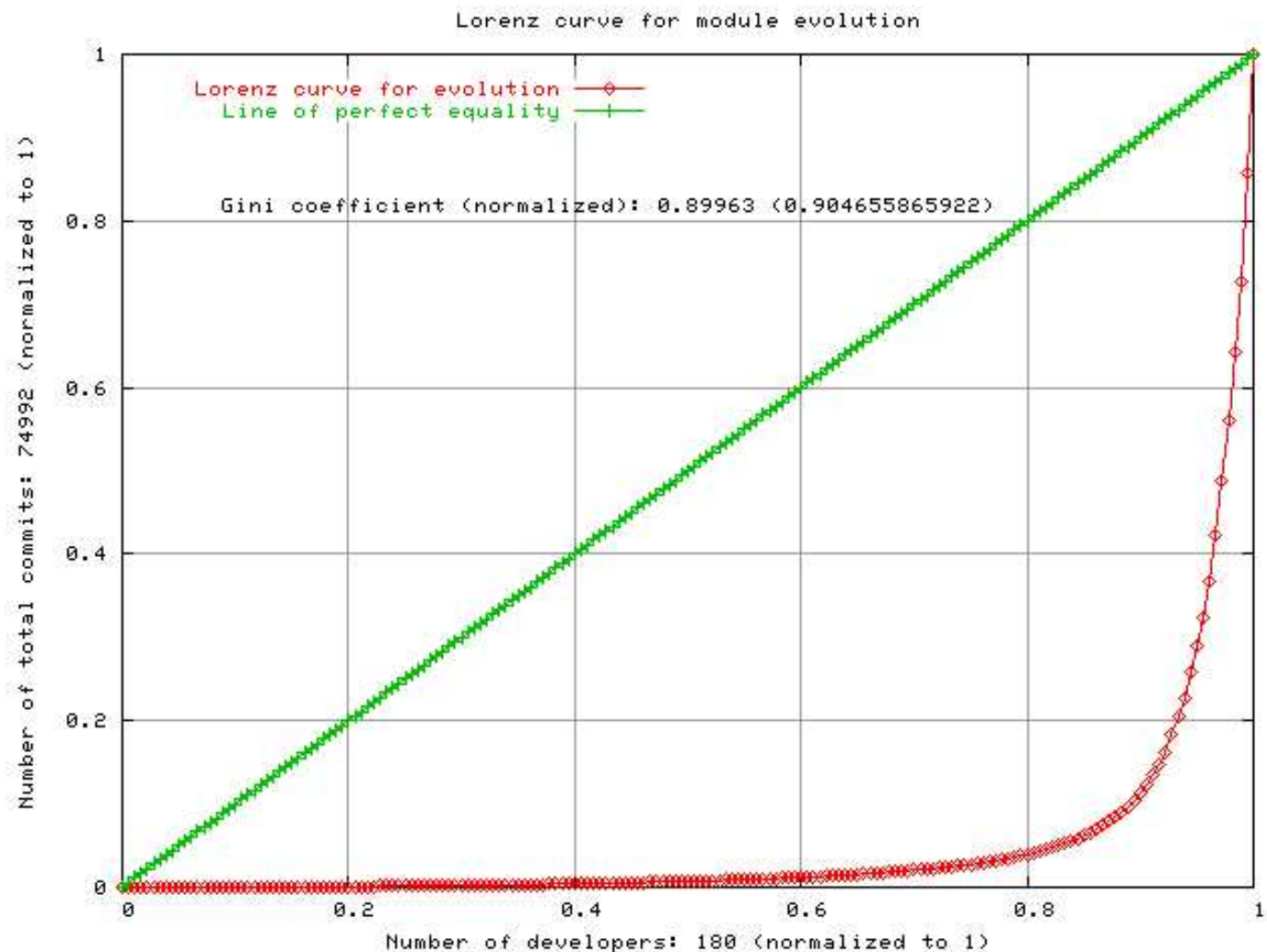
Gráfico:

- Recta: Ideal: todos los desarrolladores contribuyen igual.
- Curva: Real: Una minoría hace casi todo el trabajo.

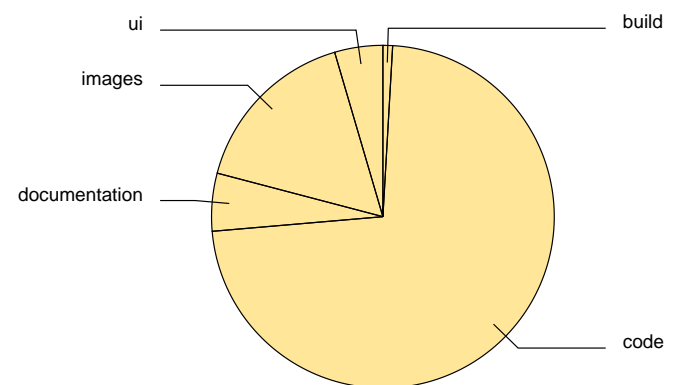
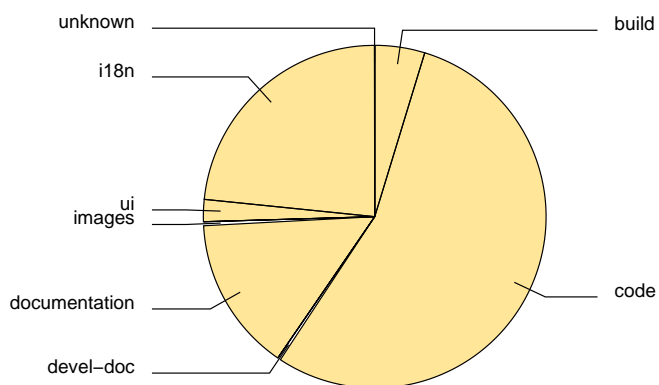
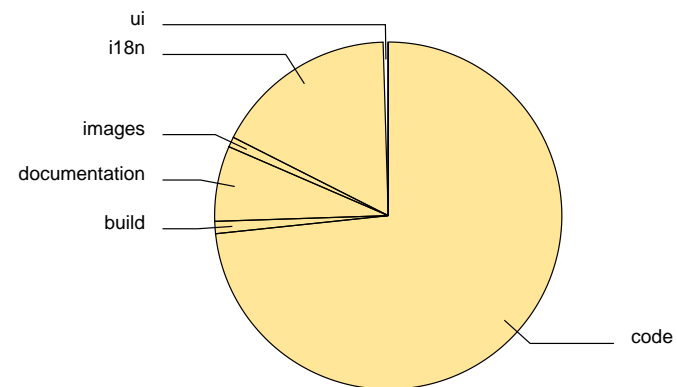
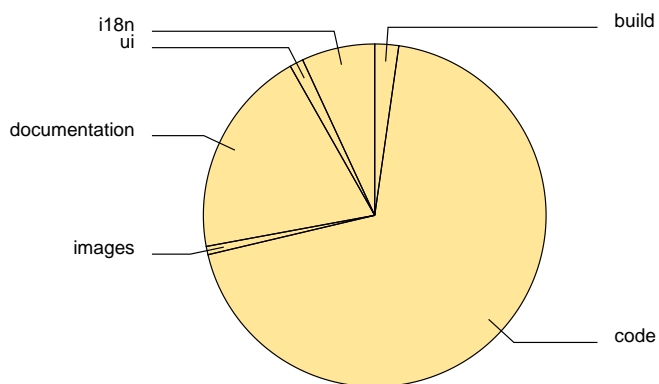
Gini: Relación entre las áreas entre recta y curva de Lorenz. (máxima desigualdad: 1).

Ley de Pareto: El 20 % de los desarrolladores realizan el 80 % del trabajo.

# Desigualdad en Evolution



# CVS: Contribuciones de cada desarrollador



## CVS: Generaciones de desarrolladores

El liderato en Evolution no es constante:

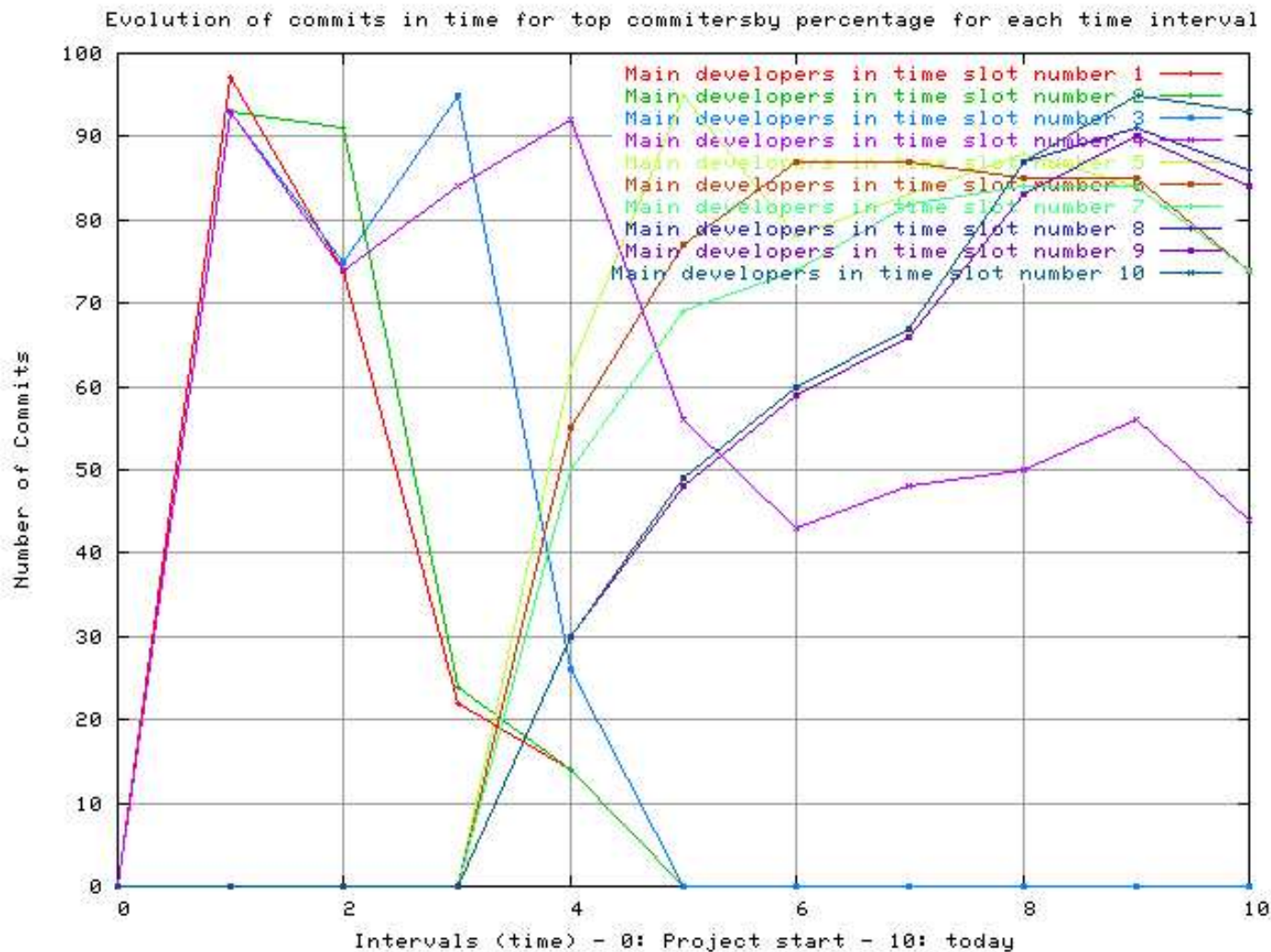
- Se va heredando entre grupos de desarrolladores (generaciones).
- No hay dependencia de un único desarrollador.

La hipótesis del “code-god” no es válida en muchos otros proyectos de software libre.

Gráfica:

- División del tiempo en intervalos.
- Identificación del grupo de desarrolladores más activo.
- Contribuciones del grupo.

# Generaciones en Evolution



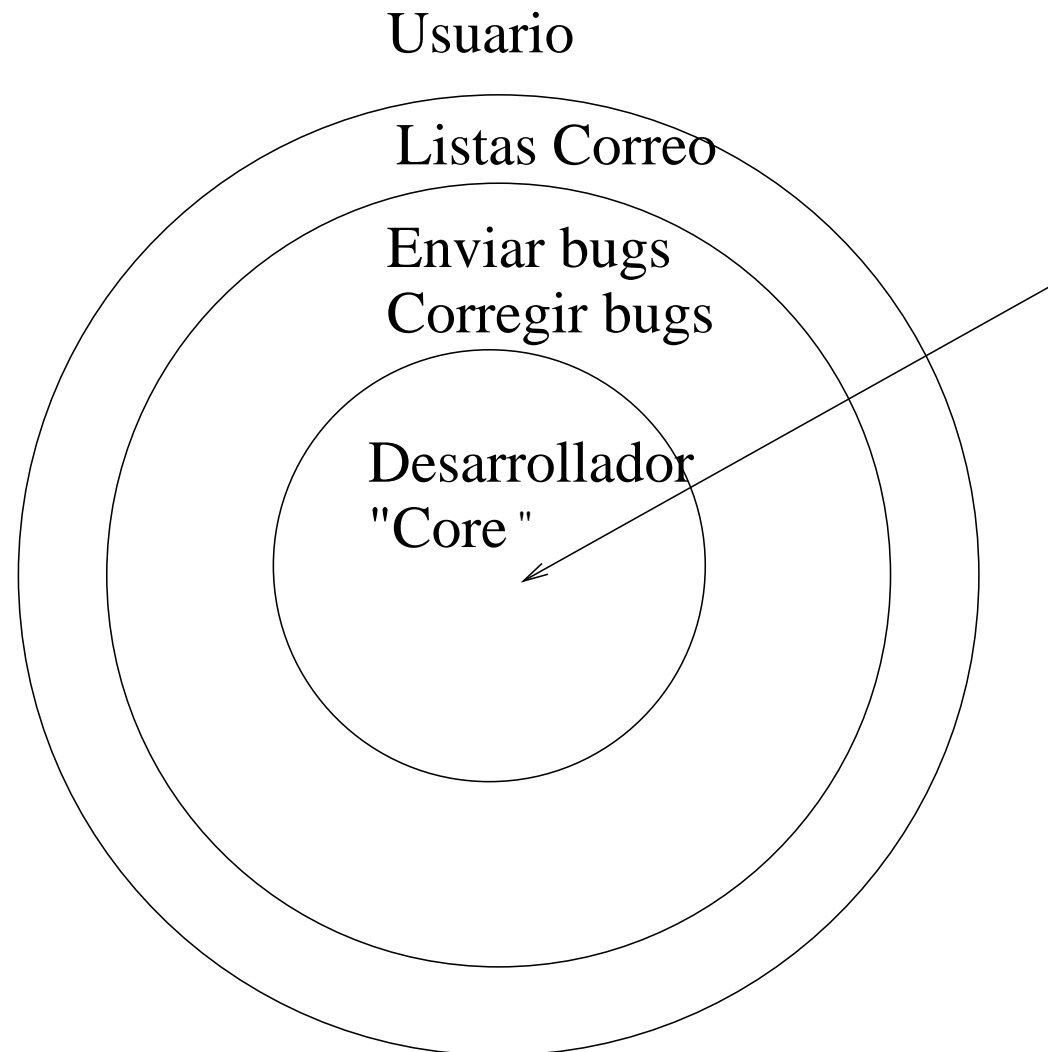
## Fuentes de datos simultáneas

- Conclusiones con varias fuentes de datos.
- Ejemplo: CVS, listas de correo, bugzilla.
- Con ellos se prueba el “modelo cebolla” en GNOME.

“Modelo cebolla” es un modelo de incorporación de desarrolladores que se ha demostrado empíricamente con GNOME.

Se trató de seleccionar desarrolladores que cumplan criterios (p.e. tiempo de permanencia como committers del CVS) y trazar su participación en listas de correo, bugzilla y CVS.

# Modelo cebolla



## Modelo cebolla

Cada desarrollador se incorpora a GNOME por “capas”:

- Inicios: usuario “pasivo”, que visita la web del proyecto y ocasionalmente las listas de correo.
- Participación en listas de correo.
- Participación en listas y en bugzilla (enviando partes).
- Contribución a la resolución de partes.
- Acceso de escritura al CVS.
- Colaboración intensa: pertenencia al grupo central de desarrolladores (“core”).

La actividad del desarrollador no puede trazarse hasta que empieza a participar al menos en las listas de correo.

## Demostración

<http://libresoft.urjc.es/>

## Conclusiones

- Necesidad de estudiar ingeniería software libre.
- Dificultades: Distribución de desarrolladores. Trabajo voluntario.
- Ventajas: Disponibilidad de gran cantidad de datos.
- Análisis mostrados: Código fuente: esfuerzo,
- CVS: desigualdad, contribuciones, generaciones,
- E-mail, BTS: Proceso gradual de integración en el proyecto.

*GNOME es una fuente de datos abundantes digna de estudio para la ingeniería del software libre.*

**Muchas gracias**

*¡Esto es todo, amigos!*