

# DeTraS

Herramientas para la investigación en la actividad de los desarrolladores

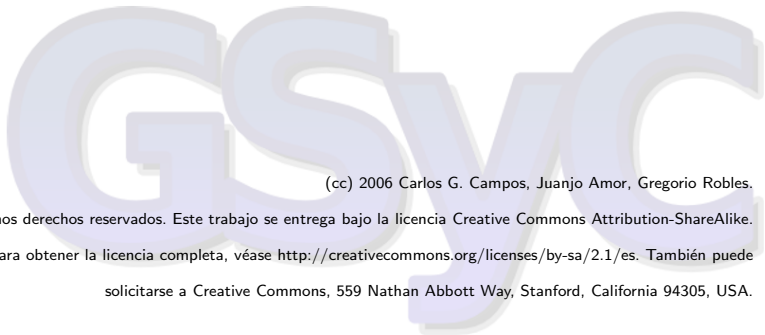
Carlos G. Campos, Juanjo Amor, Gregorio Robles

{carlosgc,jjamor,grex}@gsyc.escet.urjc.es  
GSyC/Libresoft

24 de junio de 2006



Universidad  
Rey Juan Carlos



(cc) 2006 Carlos G. Campos, Juanjo Amor, Gregorio Robles.

Algunos derechos reservados. Este trabajo se entrega bajo la licencia Creative Commons Attribution-ShareAlike.

Para obtener la licencia completa, véase <http://creativecommons.org/licenses/by-sa/2.1/es>. También puede

solicitarse a Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

# Resumen

- ¿Por qué?: Introducción y Justificación.
- ¿Cómo?: DeTraS, la herramienta.
- Conclusiones

GSyc

# Ingeniería del Software Libre

## Ingeniería del Software (Libre):

- Busca comprender el proceso de desarrollo de software
- Software Libre, gran cantidad de datos:
  - Control de versiones (CVS, SVN)
  - Listas de correo-e (Mailman)
  - Seguimiento de fallos (Bugzilla)
  - ...etc ...
- Comprender el proceso de desarrollo de software a partir de datos *empíricos*.
- Estimación de Esfuerzo, un campo de la Ingeniería del Software.

# Estimación de Esfuerzo

Estimación de costes, consiste en:

- Estimar recursos
- Estimar tiempo
- Es un **desafío constante** en la Ingeniería del Software.
- Se han desarrollado diversos modelos.
- La métrica principal es el tamaño del producto.

Ejemplos: SLIM, COCOMO [II], ...

# Esfuerzo real

Esfuerzo humano, invertido en:

- Desarrollar código
- Documentar
- Escribir correo-e
- Reuniones
- Manejo de herramientas de gestión
- Tomar café (esto mejor no lo consideramos esfuerzo ;-)

Software Libre, podemos medir actividades, entre otras:

- De producción de código fuente: CVS, SVN
- De reuniones “virtuales”: listas de correo, chats, ...
- De gestión de herramientas: Bugzilla, ...

## Esfuerzo y actividad

- El esfuerzo (y el coste) dependen de la actividad:

$$\text{esfuerzo} = g(\text{actividad}) \Rightarrow \text{coste} = f(g(\text{actividad}))$$

- Si todo fuera más sencillo:

$$\text{effort} = g(\text{activity}) = a \cdot h(\text{CVS}) + b \cdot j(\text{ML}) + c \cdot k(\text{BTS}) + d$$

- ¿Cómo definimos las funciones  $h$ ,  $j$ ,  $k$  y demás variables?
- Solución: Necesitamos conocer el esfuerzo (tiempo) que supone cada actividad en el CVS, en ML o en BTS.

## Conociendo esfuerzo

En modelos clásicos:

- El modelo podría ser similar a:  $a \cdot h(SLOC) + d$
- Se recurre a costes reales.
- Se rellenan hojas de actividad, ...

En software libre:

- Dificultad para medir esfuerzo de desarrolladores voluntarios.
- La actividad se mide con precisión (CVS, etc).
- ¿Podemos medir el esfuerzo desarrollado con precisión?

*DeTraS pretende medir el esfuerzo real invertido por los desarrolladores.*

## Conclusiones de todo esto

- ¿Son aplicables los modelos clásicos en el software libre?
- ¿Podemos aportar algo a la estimación de esfuerzo?
- ¿Podemos contar con la comunidad de desarrolladores?

GSysC

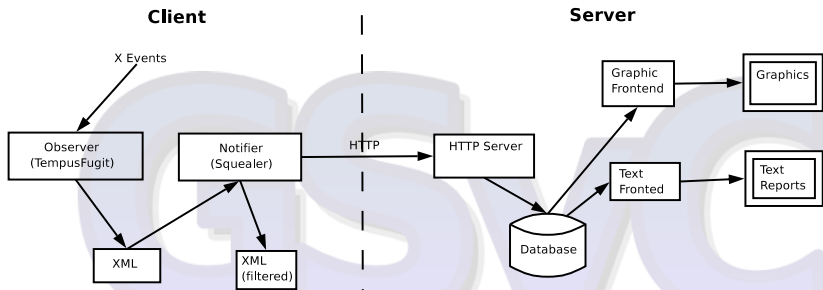
# ¿Qué es DeTraS?

- Un conjunto de herramientas para trazar la actividad de los desarrolladores
- Basado en la idea de Timeline, creado por Nat Friedman, pero escrito desde cero.
- Independiente del sistema de escritorio (en la medida de lo posible: X-window, glib, python)
- Sistema completo de seguimiento, no solo un observador de eventos

# Estructura

- TempusFugit (lado del cliente): Demonio observador
- Squealer (lado del cliente): Script que filtra y envia los datos generados por TempusFugit a nuestro servidor
- Servidor (lado del servidor): Servicio web que recibe los datos de los clientes (sequealers). Los datos son parseados y almacenados en una base de datos.
- Frontends (lado del servidor): Generadores de informes de la información almacenada en la base de datos.

# Scenario



# TempusFugit

- Observador que registra eventos que puedan estar relacionados con actividad:
  - Cambio en el foco de la ventana
  - Información del gestor de sesiones
  - Periodos de inactividad
- Escrito en C usando GObject como sistema de objetos.
- Applet del panel para el control (arrancar/parar)

# Eventos

```
<tempusfuit>
  <event time='1150402086' type='Session' event='Init' />
  <event time='1150402194' type='WindowChange'>
    <app>Evolution-2.8</app>
    <title>Correo - Evolution</title>
  </event>
  .....
  <event time='1150406883' type='Session' event='End' />
</tempusfuit>
```

# Squealer

- Script python que obtiene los datos generados por el TempusFugit, los filtra y los envía a nuestro servidor a través de un servicio web.
- Sistema de filtros basado en configuración del usuario
- Nombre de cliente único para cada desarrollador
- Sistema de configuración.

# Servidor

- Servicio web (SOAP) que permanece a la espera de recibir datos de los clientes.
- Para cada petición, parsea la información del usuario y de la traza y lo almacena en la base de datos.

# Servidor

- Datos recibidos:

- Metadatos:

```
<client id="11a41s2d1sa7070">  
  <name>Carlos García Campos</name>  
  <project>GSyC/Libresoft</project>  
  <project>GNOME</project>  
</client>
```

- Trazas:

```
<tempusfuit>  
  <event time=""1150402086"" type=""Session"" event=""Init"" />  
  <event time=""1150402194"" type=""WindowChange"">  
    <app>Evolution-2.8</app>  
    <title>Correo - Evolution</title>  
  </event>  
  .....  
  <event time=""1150406883"" type=""Session"" event=""End"" />  
</tempusfuit>
```

## Problemas a resolver

- El desarrollador debe tener el control sobre el sistema, siendo consciente en todo momento de los datos que son enviados
- Heurísticas que nos permitan diferenciar tareas de desarrollo de las que no lo son
- Configuración de usuario precisa.
- Identificador de cliente único
- Privacidad
- Notificaciones: mail

# Conclusiones

Estimación de esfuerzo:

- Una de las ramas de investigación de mayor interés empresarial
- ¿Podemos mejorar los modelos existentes?
- ¿Es el desarrollo de software libre más eficiente?

DeTraS:

- Necesidad de la herramienta.

# ¿Preguntas?

¿Preguntas?

