UNIVERSIDAD
REY JUAN CARLOS

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA INFORMÁTICA

Curso Académico 2009/2010

Proyecto Fin de Carrera

Mejoras a DeTraS: describiendo la actividad humana frente al ordenador

Autor : Edmundo Álvarez Jiménez
Tutor : Gregorio Robles Martínez

A mis padres y hermana.

Agradecimientos

Desde el primer momento en que usé un ordenador —allá por 1995—, descubrí una verdadera pasión y un buen entretenimiento. Pese a no tener grandes conocimientos en la materia, convencí a mis padres para comprarme un ordenador que me ayudase en las tareas del instituto —y, de paso, en los ratos de ocio—, con el que fui aprendiendo poco a poco de forma autodidacta. Pese a no tener muy claro la carrera que deseaba hacer hasta después de pasar la selectividad, la satisfacción que me provocaba «trastear» con el ordenador hizo que me decantase por dedicarme a la informática, decisión de la que no me arrepiento después de estos años.

Este proyecto no podría haberlo completado de no ser por la ayuda y confianza prestada por Gregorio Robles, que ha sido un buen guía durante los meses que ha durado la elaboración del proyecto, así como un gran profesor en las ocasiones que he tenido la fortuna de recibir sus clases. Tengo que agradecer también a todos los profesores que he tenido a lo largo de la carrera, por su esfuerzo, dedicación y lo mucho que me han ayudado a aprender.

Por supuesto, no podía dejar pasar esta oportunidad para agradecer a mis padres por todo el esfuerzo que han realizado para ver llegar este momento y la ayuda que siempre me han brindado. También quiero agradecer especialmente a mi hermana por todo el apoyo y cariño con el que me obsequia.

Gracias al resto de mi familia por acompañarme durante estos años. Asimismo, quiero agradecer a mis abuelos y aquellos que ya no están entre nosotros por sus enseñanzas y el tiempo que compartimos.

Finalmente, aunque no menos importante, he de dar las gracias a todos mis amigos, tanto los que me han acompañado desde hace tiempo como los que he hecho este tiempo en la universidad, por haberme obsequiado con su amistad y confianza, algo que espero mantengamos durante muchos años.

Resumen

El ser humano, desde su existencia, siempre ha necesitado medir el tiempo y conocer a qué dedica el mismo. En la actualidad, cada vez pasamos más tiempo frente a ordenadores, donde no siempre somos conscientes del tiempo que dedicamos a las distintas actividades que llevamos a cabo en ellos, al efectuar buena parte de ellas de forma rutinaria. Además, para la gestión de proyectos software esto supone otro problema, puesto que es muy complejo poder realizar una estimación de coste realista si no tenemos realimentación de otras estimaciones realizadas anteriormente, por lo que se hace necesaria una herramienta que permita dilucidar si la estimación fue realizada correctamente o no.

Para solventar los problemas comentados existen varias herramientas, siendo una de ellas DeTraS. DeTraS es un sistema comenzado a desarrollar por Carlos García Campos como Proyecto Fin de Carrera, que permite registrar los eventos que ocurren en el ordenador del usuario, y enviarlos a un servidor para su posterior análisis y tratamiento.

El presente proyecto pretende realizar una serie de mejoras y ampliaciones en el sistema DeTraS, ofreciendo un producto más sencillo de manejar, útil y funcional. Para conseguirlo, se va a trabajar en varios aspectos del sistema, mejorando la detección de inactividad, de modo que se ofrezcan resultados más realistas; aumentando la privacidad de los usuarios de la aplicación, al permitir filtrar la información que suben al servidor; actualizando y aumentando las funcionalidades ofrecidas por el *applet* de GNOME; y desarrollando una nueva herramienta que permita obtener información útil de una forma sencilla a partir de los datos recopilados en el servidor.

A lo largo de este proyecto, se ofrece una visión más amplia del problema que se pretende resolver, se presentan distintas alternativas que han surgido para resolver ese problema y se pormenoriza todo el proceso de desarrollo seguido para conseguir las mejoras mencionadas con anterioridad.

Índice general

Índice de figuras

Índice de tablas

Índice de listados

Capítulo 1

Introducción

Antes de comenzar a estudiar el presente proyecto, es importante realizarse una pregunta: ¿qué idea está detrás del mismo? La respuesta a semejante cuestión es simple a la vez que compleja: reduciéndolo a la mínima expresión, se pretende conocer a qué dedican el tiempo las personas cuando usan un ordenador. Digo que la respuesta es simple, porque todos aquellos que usamos un ordenador conocemos —o creemos conocer— qué tareas realizamos cuando nos sentamos frente al mismo. No obstante, para dar una respuesta a la pregunta formulada es preciso conocer datos objetivos, con un mínimo de rigor y disponer de un gran volumen de información que permita obtener unas conclusiones aceptables.

Aunque la idea expuesta anteriormente se escapa de los objetivos reales de este proyecto, es importante, para comprenderlo, conocer el contexto en el que surge, así como algunos enfoques que se tomaron con anterioridad a la hora de tratar este problema en distintos ámbitos. Asimismo, al ser DeTraS un proyecto de software libre, se abordará su definición y proceso de desarrollo para poder comprender la evolución del proyecto con mayor facilidad.

1.1. Estimación de esfuerzo

Uno de los mayores problemas a los que se enfrentan los gestores de software al planificar un nuevo proyecto informático consiste en la estimación del trabajo a realizar, los recursos necesarios y el tiempo transcurrido hasta su conclusión. Este proceso de evaluación es conocido dentro del campo de la ingeniería del software como estimación de esfuerzo o estimación de costes.

Hemos de tener en cuenta que, a diferencia de lo que ocurre en otras industrias,

en las que se puede conocer con bastante exactitud el coste de un producto a partir del coste de las piezas que lo componen y el tiempo necesario para su creación, siendo estos datos comúnmente conocidos, en el mundo del software no ocurre lo mismo. En este caso, únicamente es posible realizar una estimación del esfuerzo que llevará la construcción de un sistema, algo que determinará en gran medida el coste del mismo.

A la hora de realizar una estimación de esfuerzo siempre estamos asumiendo cierto grado de incertidumbre, que variará en función de diversos factores, siendo los más determinantes: la complejidad del problema, el tamaño del software y la estabilidad de los requisitos.

Evidentemente, a pesar de que el grado de incertidumbre del proyecto no sea muy elevado, sigue siendo muy complicado conocer cuánto esfuerzo va a requerir desarrollar un determinado sistema antes de construirlo, puesto que hay muchos elementos que intervienen, a cual más difícil de cuantificar, como son la experiencia, la organización del trabajo, la habilidad de los desarrolladores, el acierto en la dirección del proyecto, etcétera.

Lógicamente, cuanto mayor tiempo se dedica al desarrollo de un proyecto, más información tendremos sobre él, haciendo que el margen de error de las estimaciones sea menor. Por supuesto, aunque pudiésemos conseguir una estimación perfecta una vez hayamos terminado el proyecto, la principal utilidad de la estimación de esfuerzo es la de poder evaluar a priori, y con una exactitud aceptable, el esfuerzo que va a requerir el desarrollo del proyecto.

Pese a la inexactitud inherente a la estimación de esfuerzo, es de vital importancia para el proyecto realizar dicha valoración de la forma más ajustada posible, ya que nos permitirá reducir costes, así como aumentar los niveles de calidad, algo que puede establecer la diferencia entre obtener beneficios o pérdidas. Para conseguir ese objetivo, es indispensable la experiencia y el tener acceso a una buena información histórica, es decir, conocer datos sobre anteriores proyectos.

Como cabe esperar en un proceso tan complejo, muchas personas han intentado solventar los problemas que entraña la estimación de esfuerzo en la ingeniería del software, ideando para ello un buen número de técnicas con mayor o menor complejidad y éxito. A continuación se presentarán y analizarán brevemente algunas de las técnicas más conocidas e interesantes.

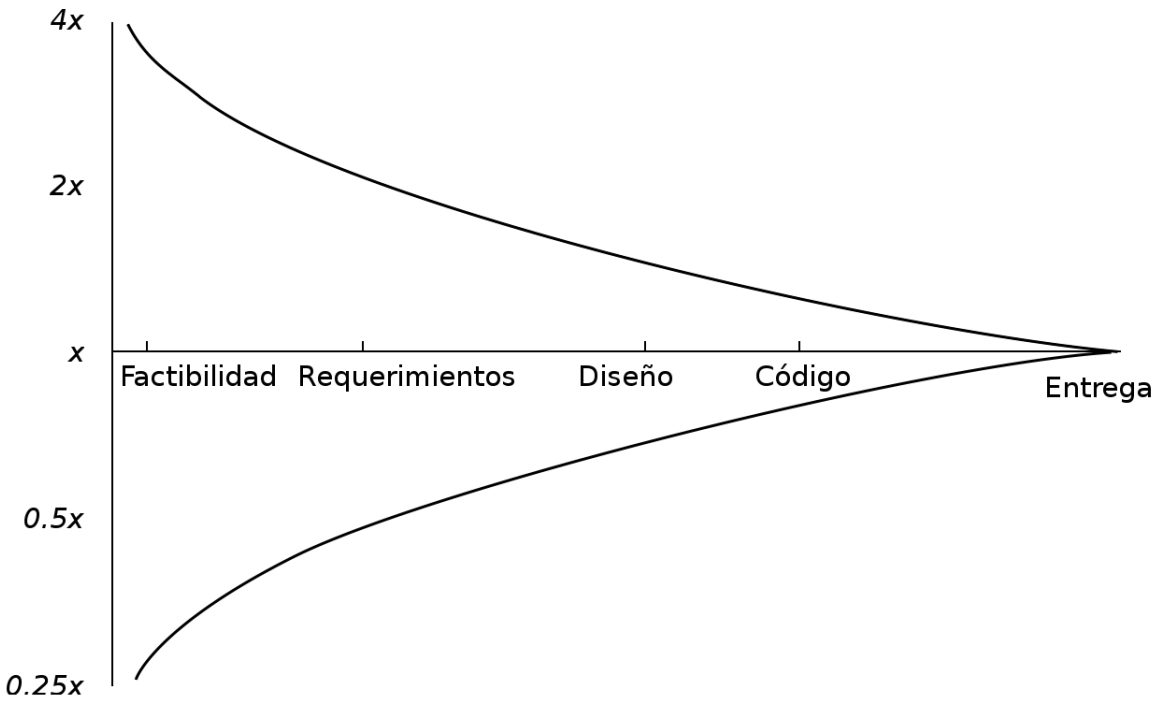

Figura 1.1: Incertidumbre estimada.

1.1.1. Juicio experto

Muchos métodos de estimación se basan en la experiencia de gerentes de proyectos semejantes al que se desea desarrollar. De este modo, si el sistema a desarrollar A es similar al sistema desarrollado B, el esfuerzo requerido para producir A debería ser parejo al de producir B.

Este proceso puede formalizarse solicitando a varios expertos que estimen el esfuerzo de forma pesimista, optimista y más probable. Una vez han sido evaluadas las estimaciones de los expertos, se realiza la media de la distribución beta de probabilidades determinada por esos valores, obteniendo una estimación normalizada. La expresión que define este valor normalizado es la siguiente:

$$\frac{(x + 4y + z)}{6},$$

donde x representa la estimación pesimista, y la optimista y z la más probable.

La técnica Delphi emplea el dictamen de los expertos de una forma más sofisticada. Se solicita a cada experto que efectúe una estimación individual, basada en su experiencia, de forma confidencial. Posteriormente, se calcula el promedio de las esti-

maciones y se muestra su valor a los expertos, dándoles la posibilidad de modificar su anterior valoración. Este proceso se repite hasta llegar a un consenso, siendo todas las estimaciones secretas e individuales durante el mismo.

Los modelos basados en la opinión de expertos dependen directamente de su habilidad para determinar qué proyectos están relacionados y cómo afectan las posibles diferencias existentes entre ellos al esfuerzo necesario. Por tanto, las estimaciones obtenidas empleando estos modelos serán muy subjetivas y posiblemente imprecisas.

1.1.2. Modelo de Bailey–Basili

Bailey y Basili sugirieron en 1987 una técnica de modelado para realizar una ecuación de estimación que reflejase las características de una organización.

Demostraron su técnica utilizando datos de dieciocho proyectos científicos de gran tamaño, obteniendo una ecuación muy precisa:

$$E = 5,5 + 0,73S^{1,16},$$

donde S es la dimensión estimada del sistema en líneas de código.

A su vez, Bailey y Basili explicaron otros factores que afectan al esfuerzo, y que permiten ajustar la ecuación anterior. El proyecto debe ser puntuado entre cero —ausente— y cinco —muy importante— para cada factor indicado en la tabla 1.1, dependiendo de la opinión del gerente del proyecto.

1.1.3. SLIM

Es un método de estimación creado por Lawrence Putnam —de ahí que también sea conocido como método de Putnam—, que muestra la distribución del esfuerzo en relación al tiempo. Es aplicable a proyectos muy grandes —más de setenta mil líneas de código—, aunque es posible ajustar las expresiones para proyectos más pequeños.

SLIM se basa en que la relación entre el esfuerzo requerido y el tiempo necesario para desarrollar un proyecto software no es lineal, sino que es descrita por curvas de Rayleigh. Esto implica que pequeños cambios en el tiempo de desarrollo pueden provocar grandes modificaciones en el esfuerzo requerido.

El elemento central del método de Putnam es la «ecuación del software»:

$$Producto = PP * \left(\frac{Esfuerzo}{B}\right)^{\frac{1}{3}} * Tiempo^{\frac{4}{3}},$$

Metodología total	**Complejidad acumulada**	**Experiencia acumulada**
Diagramas de árbol	Complejidad de la interfaz del cliente	Calificación de los programadores
Diseño descendente	Complejidad de la aplicación	Experiencia de máquina de los programadores
Documentación formal	Complejidad del flujo del programa	Experiencia de los programadores con el lenguaje
Equipos de programadores jefe	Complejidad de la comunicación interna	Experiencia de los programadores con la aplicación
Entrenamiento formal	Complejidad de la base de datos	Experiencia del equipo
Planes de prueba formales	Complejidad de la comunicación externa	
Formalismos de diseño	Cambios al diseño de programas iniciados por el cliente	
Lectura del código		
Carpetas de desarrollo de unidades		

Tabla 1.1: Modificadores del esfuerzo según Bailey-Basili.

donde *Producto* representa cierta medida sobre la funcionalidad del mismo y se mide en líneas de código; PP es el *Parámetro de Productividad*, una constante que engloba los factores del entorno y expresa la productividad de la compañía; el $Esfuerzo$ comprende las distintas etapas por las que pasa el proyecto y se mide en personas por año; B es un parámetro que representa la destreza en función del tamaño del sistema; y el *Tiempo* requerido para completar el desarrollo del software medido en años. Las potencias empleadas en la expresión anterior, fueron obtenidas de forma empírica a mediados de los 80, utilizando datos de 750 sistemas.

La ecuación del software debe estimar el tiempo y el esfuerzo de desarrollo, que son dos incógnitas de la expresión. Además, es necesario conocer el PP de la organización mediante proyectos anteriores y una estimación de las líneas de código del producto.

Para poder estimar el esfuerzo, Putnam propone otra expresión:

$$PIP = \frac{Esfuerzo}{Tiempo^3},$$

donde PIP es un *Parámetro de Incremento de Personal*, valor que se conoce al principio del proceso.

La conclusión que puede extraerse de esta segunda expresión es que el aumento rápido en el personal de desarrollo tiene más efecto en el coste y el esfuerzo que en el tiempo.

Combinando las expresiones anteriores podemos conseguir otra en la que el tiempo no esté presente:

$$Esfuerzo = \left(\frac{Producto}{PP}\right)^{\frac{9}{7}} * PIP^{\frac{4}{7}}$$

1.1.4. Método COCOMO

Uno de los problemas de los métodos vistos en los apartados 1.1.2 y 1.1.3 es su dependencia del tamaño del sistema. Las estimaciones suelen ser necesarias al comienzo, cuando no se dispone de información suficiente para determinar con precisión las dimensiones del sistema. De esta forma, este tipo de modelos terminan intercambiando el problema de estimar el esfuerzo por el problema de estimar el tamaño del sistema. El modelo COCOMO en su versión más reciente reconoce este problema y trata de resolverlo.

El método COCOMO —cuyo nombre es un acrónimo de ***CO**nstructive **CO**st **MO**del* o Modelo Constructivo de Coste— fue propuesto por Barry Boehm en 1981, y es uno

de los modelos de estimación de coste de software más empleados y estudiados. El modelo original ha sido revisado, obteniendo en 1995 una segunda versión del mismo —llamada COCOMO II—, siendo mucho más completa que el modelo original. De ahora en adelante me centraré en esta última versión, al ser la más actual y perfeccionada.

Siendo precisos, COCOMO más que un modelo es una jerarquía de modelos de estimación, es decir, se proponen modelos distintos para las distintas situaciones que atraviesa un proyecto software, ofreciendo así una mayor fidelidad en cada una de ellas. Estos tres modelos son:

Modelo de composición de la aplicación

Se dirige a las primeras etapas de desarrollo, cuando no se conocen con detalle los requisitos de la aplicación y se desarrollan prototipos, a partir de componentes ya existentes, de la interfaz de usuario, la interacción del sistema, etcétera. Como se puede intuir, en estas etapas tan tempranas de desarrollo no se tiene información sobre las dimensiones que tendrá el sistema, por lo que COCOMO II estima el mismo en función de generadores de esfuerzo de alto nivel, como el número de pantallas e informes, que se conocen con el nombre de puntos de aplicación.

Modelo de diseño anticipado

Es empleado cuando los requisitos se estabilizan y se tiene una idea —aunque puede que no sea definitiva— de la arquitectura del software. En este momento se conoce más información para realizar una estimación que en el punto anterior, aunque aún no es suficiente para realizar una valoración precisa. Por este motivo, en lugar de usar puntos de aplicación o líneas de código como medida de tamaño, se emplean puntos de función. Los puntos de función son una técnica que estima la funcionalidad capturada en los requisitos ofreciendo una descripción del sistema más rica que los puntos de aplicación.

Modelo *postarquitectonico*

Este modelo se orienta al momento en el que el desarrollo comienza y se tiene mucha más información que permite realizar una estimación razonablemente precisa del

tamaño del software. En este modelo es posible dimensionar el sistema mediante puntos de función o líneas de código, pudiendo incluirse factores de costo que reflejan la capacidad del personal, el producto y las características del proyecto.

1.1.5. Método de caracterización de la actividad del desarrollador

El método de caracterización de la actividad del desarrollador es fruto del estudio de numerosos proyectos de software libre realizado por tres miembros de esta universidad: Juan José Amor, Gregorio Robles y Jesús M. González–Barahona. Es un método muy reciente, para ser precisos data del año 2006, y surge como una alternativa a los métodos de estimación de esfuerzo tradicionales, con el objetivo de adaptarse mejor a los proyectos de software libre, aunque no es exclusivo para ellos.

En lugar de basar la estimación en el número de líneas de código, como hacen la mayor parte de los métodos de estimación de esfuerzo existentes, sus autores piensan que es posible mejorar la estimación de costes considerando otras métricas basadas en la caracterización de las actividades realizadas por los desarrolladores, es decir, analizando su comportamiento durante el proceso de desarrollo.

Hay que tener en cuenta que en el desarrollo de software «tradicional», existe un grupo fijo de desarrolladores que tienen un horario muy ajustado y deben dedicar un determinado número de horas semanales a sus tareas. Sin embargo, esto no ocurre así en el desarrollo de software libre, ya que es habitual que colaboradores externos al proyecto intervengan en el desarrollo del mismo y dediquen un número indeterminable de horas a esa labor. Los sistemas de estimación de esfuerzo tradicionales fallan en este punto, porque no tienen en cuenta la intervención de esos colaboradores externos y, aunque esas personas no formen parte del equipo de desarrollo, su participación también influye en el esfuerzo final requerido para completar el software.

El método de caracterización de la actividad del desarrollador propone evaluar el coste de un proyecto con la siguiente expresión:

$$coste = \sum coste_{interno} + \sum coste_{desarrolladores\_externos}$$

Asimismo, establece que el coste es función del esfuerzo requerido, y este es, a su vez, función de las actividades realizadas por el desarrollador. Por tanto, es posible estimar el esfuerzo de producción del software a partir de los datos producidos en un proyecto libre —que son detallados en el apartado 1.4.2—.

1.2. Administración del tiempo

El tiempo es un recurso muy valioso que no podemos almacenar, detener o adquirir. Por este motivo, habitualmente nos preocupamos por el tiempo que dedicamos a realizar las tareas que llevamos a cabo —ya sea en el mundo laboral o en el personal—, e intentamos organizarlas y gestionarlas para conseguir emplear mejor el tiempo. Lo más frecuente es que cada uno se organice «a su manera»: establecer un horario indicando y separando distintas actividades, planificar las tareas a realizar y anotarlas en una lista, etcétera. Puesto que estos métodos no siempre son efectivos, han surgido distintas técnicas que prometen organizar mejor nuestra forma de realizar actividades, con el fin de poder utilizar el tiempo de una forma más eficaz. Entre estas técnicas destacan: Getting Things Done[1], los siete hábitos de Stephen Covey[2], Zen To Done[3] y Pomodoro[4].

Al pasar cada vez una mayor cantidad de tiempo usando ordenadores, es interesante conocer cuanto empleamos en las distintas tareas que realizamos frente ellos. Esto tiene una especial relevancia cuando la persona se dedica al desarrollo de software libre, ya que en muchos casos es desarrollado por personas en su tiempo libre.

1.3. Estadísticas de uso de aplicaciones

La necesidad de saber cuáles son los productos o servicios más usados en cualquier campo hace que se efectúen estadísticas con el objetivo de conocer cuál es su impacto frente a la competencia. Algunos ejemplos de estos estudios podemos observarlos en el cálculo y comparación de la audiencia televisiva, del número de abonados en las compañías de telefonía móvil, de la difusión de la prensa, etcétera.

El caso del software no es una excepción, y son datos que interesan a muchas partes:

- Las empresas desean conocer estadísticas de uso de sus aplicaciones para poder compararse con otras, con el objetivo de convencer a más personas de usar sus productos, conseguir mejores contratos y observar cómo evolucionan los intereses de los usuarios. Dos de los sectores con mayor competencia en la industria de

[1] http://es.wikipedia.org/wiki/Getting_Things_Done

[2] http://en.wikipedia.org/wiki/The_Seven_Habits_of_Highly_Effective_People

[3] http://zenhabits.net/2007/04/zen-to-done-ztd-the-ultimate-simple-productivity-system

[4] http://www.pomodorotechnique.com

la informática, como son los navegadores web y los sistemas operativos, disponen de sus cuotas de mercado actualizadas frecuentemente.

- Los desarrolladores independientes —por ejemplo, de software libre— también pueden estar interesados en conocer este tipo de estadísticas, aunque para ellos suele ser menos relevante.

- Los desarrolladores de sistemas operativos desean conocer los programas más empleados para incluir aplicaciones similares en ellos. Esto es especialmente patente en las distribuciones de GNU/Linux.

- Los administradores de sistemas desean conocer las aplicaciones usadas en los sistemas bajo su responsabilidad con el fin de instalar lo que sea más usado, eliminar programas que no se empleen, etcétera.

- Algunos usuarios también están interesados en conocer qué aplicaciones utilizan otras personas, con diversos objetivos, aunque destacando el conocer nuevas —y puede que mejores— aplicaciones.

Pese a que, como podemos observar, estas medidas resultan bastante interesantes, en el caso del software es bastante complejo conocer las aplicaciones que utiliza cada persona, principalmente a causa de las distintas plataformas empleadas por los usuarios, la enorme diversidad de aplicaciones existentes y sus formas de distribución. Por ello, se realizan distintas estadísticas en función de diversos factores: medir las descargas efectuadas de su programa, contabilizar las licencias vendidas, evaluar el número de visitas a determinadas páginas web, o estudiar las búsquedas realizadas en Internet.

1.4. Software libre

Para aclarar el concepto de software libre emplearé la definición realizada por la *Free Software Foundation*[5] [10]:

> *El «Software Libre» es un asunto de libertad, no de precio. Para entender el concepto, debe pensarse en «libre» como en «libertad de expresión», no como en «cerveza gratis».*
>
> *«Software Libre» se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software.*

[5] *Free Software Foundation* es una organización creada con el objetivo de difundir y apoyar el movimiento del software libre. Su sitio web es `http://www.fsf.org`.

Según esta misma organización, podemos distinguir un programa libre de otro privativo por cumplir estas cuatro libertades:

Libertad 0. La libertad de usar el programa, con cualquier propósito.

Libertad 1. La libertad de estudiar el funcionamiento del programa y adaptarlo a las necesidades.

Libertad 2. La libertad de distribuir copias, con lo que se puede ayudar a otros.

Libertad 3. La libertad de mejorar el programa y hacer públicas las mejoras, de modo que toda la comunidad se beneficie.

Para que puedan cumplirse las libertades 1 y 3 es indispensable que el código fuente sea accesible.

1.4.1. Desarrollo de software libre

En la actualidad, existe un gran número de proyectos de software libre. Muchos de ellos son impulsados y mantenidos por empresas, aunque la mayoría son desarrollados por la comunidad del software libre. Esta comunidad no es más que un colectivo de desarrolladores y usuarios distribuidos geográficamente que apoyan el software libre, y que suelen colaborar en proyectos libres usando herramientas a través de Internet.

Puesto que todo el proceso de desarrollo se suele realizar a través de la Red, cada proyecto genera una gran cantidad de información que puede ser consultada por cualquiera. Entre esa información destacan las decisiones tomadas durante el desarrollo, el código fuente de la aplicación, y la información sobre los defectos que se encuentran en la misma.

Como vimos en el punto 1.1, las técnicas de estimación de esfuerzo requieren gran cantidad de información sobre el sistema en desarrollo y el entorno para poder ofrecer unas estimaciones precisas. En el modelo de software privativo suelen emplearse para obtener estos datos otros proyectos realizados en la propia empresa, al ser los únicos de los que se tiene suficiente información. Esto contrasta con la gran cantidad de proyectos libres que se pueden encontrar, haciendo que sea más sencillo dar con proyectos —o parte de los mismos— que encajen mejor con el sistema a desarrollar. Por si esto fuera poco, dispondremos además de la información generada día a día durante su desarrollo, facilitando el estudio de la aplicación.

1.4.2. Herramientas para manejar la información generada

Debido a la gran cantidad de información que se maneja en un proyecto de software libre, la necesidad de que esta información se encuentre disponible en línea, y las distintas procedencias de la misma, se han creado varias herramientas capaces de facilitar la colaboración e interacción entre los miembros de la comunidad. Algunas de estas herramientas son:

Listas de correo

Como el software libre es desarrollado generalmente por personas que se encuentran distribuidas geográficamente, es vital disponer de una herramienta de comunicación para debatir y tomar decisiones sobre el proyecto.

Las listas de correo juegan un papel fundamental en este aspecto, puesto que permiten enviar mensajes de correo electrónico a todos sus componentes, sin importar si se encuentran en línea en ese momento o el número de personas suscritas a la lista.

Generalmente, estas listas suelen ser públicas y se puede acceder a ellas vía web, por lo que toda la toma de decisiones del proyecto puede ser consultada y analizada por cualquiera que lo desee.

Sistemas de control de versiones

Básicamente, sirven para almacenar el código fuente de la aplicación, aunque tienen muchas más funcionalidades que facilitan el desarrollo de software en grupo. Algunas de estas características son: solucionar problemas de concurrencia cuando varias personas trabajan con el mismo fichero; simplificar el control de versiones de los distintos ficheros que componen la aplicación, conociendo información sobre la fecha del cambio, su autoría y los cambios realizados; permitir la creación de varias «ramas de desarrollo», pudiendo existir una versión estable y otra experimental de la misma aplicación con las que se puede trabajar independientemente; y facilitar la vuelta atrás en el código, pudiendo restaurar una versión anterior de un fichero —o varios— si se desea revertir algún cambio.

Sistemas de seguimiento de errores

Los sistemas de seguimiento de errores tienen como principal objetivo recopilar informes sobre defectos del software con el fin de ser solucionados en el futuro.

Cuando estas aplicaciones son usadas en el desarrollo de software libre, suelen ser los propios usuarios de la aplicación los que informan de los errores. Este hecho potencia el concepto de comunidad existente, ya que los desarrolladores se benefician de los reportes de errores de los usuarios, y estos ven que su aportación es tenida en cuenta en el desarrollo de la aplicación.

Normalmente, es posible establecer el grado de relevancia del problema indicado, pudiendo priorizar los mismos en función de su importancia, así como seguir el estado en el que se encuentra la solución al problema.

Además de estos usos, también suelen emplearse este tipo de sistemas para proponer nuevas funcionalidades a los desarrolladores y para gestionar las tareas pendientes de los mismos.

Como es habitual, existen varios sistemas de seguimiento de errores con filosofías diferentes, aunque lo más importante es que consiguen hacer que todo el proceso de informe y corrección de errores sea transparente y que cualquiera pueda participar en él, ya sea reportando problemas o incluso proponiendo soluciones a los mismos.

Wikis

En los últimos tiempos están tomando una gran relevancia a la hora de aportar documentación para las aplicaciones.

Un *wiki* es un mecanismo de colaboración consistente en un sitio web cuyo contenido puede ser editado a través de un navegador web. De esta forma, cualquiera —generalmente previo registro— puede aportar sus conocimientos sobre algún aspecto del proyecto libre, construyendo un sitio web con gran cantidad de documentación de forma *colaborativa*, siendo además fácilmente editable y muy dinámico.

Creo que todos en la actualidad conocemos Wikipedia[6], que no es más que un enorme *wiki* empleado como enciclopedia.

[6]Wikipedia es un proyecto de la Fundación Wikimedia para construir una enciclopedia libre y políglota. `http://www.wikipedia.org/`

Capítulo 2

Estado del arte

En el presente capítulo se va a concretar y especificar más en profundidad el problema al que se enfrenta este proyecto, exponiendo, a su vez, varias soluciones concretas surgidas para dar respuesta a esa cuestión.

Seguidamente, se describirán las tecnologías usadas para desarrollar el proyecto, sin las cuales su ejecución podría haber resultado mucho más costosa.

2.1. Descripción del problema

Aunque en principio pueda no ser evidente, los puntos detallados en la introducción tienen distintos problemas que pueden solventarse con una única aplicación.

Por un lado, hemos visto que necesitamos información histórica para conseguir realizar una valoración de esfuerzo ajustada. Para obtener estos datos podemos ayudarnos de estimaciones de proyectos desarrollados con anterioridad en la misma empresa y/o de proyectos libres, que podemos encontrar en Internet. No obstante, es complejo seguir la pista de las actividades que han ido realizando los desarrolladores durante la ejecución del proyecto, y esto nos daría una información muy útil sobre el esfuerzo realizado.

Si en los proyectos consultados empleasen las herramientas detalladas en el apartado 1.4.2, podríamos observar las acciones llevadas a cabo por cada desarrollador a través de *commits* realizados en el sistema de control de versiones, defectos resueltos que se indican en los sistemas de seguimiento de errores, etcétera. Esto nos indicaría cuándo se ha completado una tarea en el desarrollo, aunque no proporciona datos sobre cuánto tiempo ha llevado completar esa tarea, que es una información que ayudaría a que la estimación fuese más precisa.

Si nos centramos en la administración del tiempo cuando empleamos el ordenador, podemos ver que es complejo determinar con exactitud el tiempo que nos han tomado las actividades que realizamos. Podemos afirmar que hay ocasiones en las que incluso no recordamos todas las tareas que hemos llevado a cabo mientras estábamos enfrente del computador, quizá seamos capaces de evocar las actividades más importantes que hemos desarrollado, aunque es difícil acordarse de todas ellas por diversos motivos: es frecuente trabajar con varios programas a la vez, por lo que podemos realizar varias tareas simultáneamente; existen distracciones que impiden que empleemos todo el tiempo en lo que teníamos previsto; las acciones son tan rutinarias que las efectuamos casi sin percatarnos; etcétera.

Por último, también cito en la introducción algunas técnicas usadas para calcular las estadísticas de empleo de aplicaciones. Sin duda podemos afirmar que son muy variopintas, lo que provoca que sea complejo relacionar las medidas ofrecidas entre sí. Por ejemplo, para calcular el número de usuarios de Windows y Ubuntu no podemos comparar las ventas de licencias del primero con las descargas del segundo por diversos motivos. Además, en algunas técnicas, habría que tener cuidado con la forma de realizar las medidas para no desvirtuarlas. Así, no deberíamos estimar el número de usuarios de Windows y Ubuntu recogiendo datos de las personas que acceden a los foros de Ubuntu, porque muy probablemente será un sitio web más visitado por los usuarios de esa distribución de GNU/Linux.

Si ponemos todo en común, vemos que estos problemas pueden ser resueltos con una aplicación que observe las tareas que realiza el usuario en el ordenador y el tiempo que les dedica. Además de mostrar esos datos localmente, debería ofrecer la opción de enviarlos a un servidor remoto donde se pudiesen almacenar y tratar esos datos, con el fin de observar el uso de las distintas aplicaciones empleadas, así como el tiempo de uso de cada una. Esa es la idea básica que ofrece el sistema DeTraS del que hablaremos en la sección 2.4 y que este Proyecto Fin de Carrera tiene por objeto extender.

Finalmente, quiero destacar que aunque DeTraS —y probablemente otros de los métodos de los que hablaré en este capítulo— pueden resolver todos los problemas detallados en este apartado, de ahora en adelante me centraré en el problema que me resulta más interesante, que no es otro que el seguimiento de las actividades de desarrollo.

2.2. Encuestas

Una forma de conocer las actividades que realizan los desarrolladores es tan sencilla como realizar encuestas periódicamente, solicitando datos sobre las tareas desarrolladas y el tiempo que han requerido. Estas encuestas se pueden llevar a cabo de forma telefónica, en papel o empleando alguna aplicación informática.

Evidentemente, este método es muy impreciso, ya que si se realiza la encuesta cada poco tiempo —imaginemos que se efectuase una diaria—, sería muy tedioso y molesto para los desarrolladores, y probablemente la responderían sin prestarle demasiada atención. Por contra, si se realiza la encuesta más esporádicamente —pongamos una vez al mes— muchos detalles desarrollados habrían caído en el olvido, y el tiempo que se necesitó para completar las actividades sería demasiado impreciso.

Además, hay que tener en cuenta que algunos de los métodos que suelen emplearse para realizar encuestas pueden ser *intrusivos* y molestos. Por este motivo deberían descartarse las encuestas telefónicas, que son sin duda las más molestas, ya que además de interrumpir al desarrollador mientras está trabajando, le obligan a contestar a las preguntas en ese preciso momento. También debería evitarse emplear encuestas automáticas que utilicen ventanas emergentes para mostrar la encuesta o avisar de que está disponible para su cumplimentación.

2.3. Proyecto Hamster

Hamster es un *applet*[1] de seguimiento temporal desarrollado para GNU/Linux, que forma parte del entorno de escritorio GNOME[2]. Su principal funcionalidad es permitir conocer las tareas que se realizan y cuanto tiempo requieren. La aplicación, a su vez, posibilita la agrupación de actividades en categorías, facilitando el posterior análisis de la información. Además, es capaz de mostrar todos los datos recogidos empleando gráficos diarios, semanales o mensuales, de modo que sea posible analizar la información de una forma más visual.

Esta aplicación puede cumplir en parte con los requisitos para realizar un seguimiento de las actividades que efectúa un desarrollador, aunque tiene una carencia im-

[1] En el contexto de los entornos de escritorio, un *applet* es una pequeña aplicación cuya interfaz de usuario reside en un panel del mismo.

[2] `http://www.gnome.org/`

portante: todo su funcionamiento se basa en introducir manualmente la tarea que se va a realizar en el momento preciso en que se comienza a ejecutar. Esto supone un problema cuando se está trabajando debido a que el foco de atención de la persona —en este caso del desarrollador— está centrado principalmente en la tarea que debe realizar, dejando en un segundo plano otras actividades. Por tanto, es probable que el desarrollador no recuerde que debía cambiar la tarea en la aplicación, alterando los datos reales. Asimismo, en el caso de estar llevando a cabo dos tareas de forma simultánea, o actividades que requieren de poco tiempo, puede llegar a ser molesto el tener que editar cada poco tiempo la ocupación actual.

2.4. El sistema DeTraS

El sistema DeTraS —acrónimo de ***De**veloper **Tra**cking **S**ystem* o Sistema de Seguimiento de Desarrolladores— es el resultado de un Proyecto Fin de Carrera realizado en el curso 2005/2006 por Carlos García Campos. En esencia, la idea con la que este proyecto pretende resolver el problema expuesto con anterioridad consiste, según su autor [1], en:

> *. . . capturar todo lo que el desarrollador está haciendo en su ordenador, para posteriormente analizarlo y estudiarlo.*

Para llevar a la práctica esta idea, el sistema se encarga de observar las acciones que se van realizando en el ordenador, «anotando» las distintas aplicaciones que usa el desarrollador y el tiempo empleado en cada una de ellas. Además, DeTraS incluye, junto a esa información, datos sobre la actividad de la sesión del usuario, comprendiendo el inicio y cierre de sesión, así como los periodos de inactividad —entendiendo que la sesión de usuario está inactiva cuando lleva más de cinco minutos sin que se haya empleado el teclado o el ratón—. Todo ello constituye una fuente de información muy valiosa a la hora de analizar el esfuerzo del desarrollador en las distintas etapas del desarrollo software.

No obstante, las tareas del sistema DeTraS no se limitan a reunir la información indicada previamente: permite a su vez enviar la información acumulada durante el uso del programa a un servidor, con el fin de poder estudiar los datos de uno o varios usuarios detalladamente, y facilita la generación de informes con los datos almacenados en el servidor, suministrando una serie de *scripts* genéricos y una biblioteca de

programación que simplifica la creación de nuevos *scripts*.

Como se puede advertir, partiendo de la idea expuesta con antelación es posible resolver muchos de los problemas existentes en otros métodos de cálculo de esfuerzo, haciendo este proceso transparente para el desarrollador y bastante exacto.

Pese a que el sistema DeTraS es funcional y bastante interesante en su estado actual, tiene algunos aspectos que deben ser mejorados o ampliados para intentar aumentar su uso y difusión, algo que nos permitiría realizar estudios valorando el esfuerzo de desarrolladores de todo el mundo, al mismo tiempo que la comunidad de software libre se implica en la evolución del sistema. Esos son los objetivos del presente proyecto, que partirá de la base aquí detallada.

2.5. Herramientas y tecnologías utilizadas

Como puede imaginarse el lector, las herramientas y tecnologías informáticas empleadas para llevar a cabo el desarrollo del presente proyecto son muy variadas y numerosas. En este apartado se describirán brevemente las más importantes que se han utilizado, así como su relación con el proyecto. Algunos de los objetivos y componentes del software desarrollado aquí indicados se presentarán en el capítulo 3.

2.5.1. C

C [11, 12] es un lenguaje de programación creado en 1972 por Dennis M. Ritchie en los Laboratorios Bell. Se trata de un lenguaje de propósito general con una sintaxis sencilla de aprender, estructuras de datos simples pero muy versátiles y un rico conjunto de operadores.

Este lenguaje de programación está a medio camino entre los lenguajes de alto nivel y los de bajo nivel, ya que tiene una estructura y características típicas de los lenguajes de alto nivel, permitiendo a su vez realizar acciones a muy bajo nivel.

A lo largo de su vida, C ha sufrido varios procesos de estandarización —en 1989 por el organismo ANSI y en 1990 por el ISO— que permiten conseguir un código independiente de la máquina si no se emplean elementos dependientes del compilador o el sistema operativo.

Pese a no tratarse de un lenguaje de alto nivel estrictamente hablando, C tiene varias ventajas frente a estos lenguajes, erigiéndose como una muy buena opción a la

hora de desarrollar programas: es muy eficiente y pueden emplearse características de bajo nivel para optimizar la implementación; es posible emplearlo en prácticamente todos los sistemas; y permite realizar programas modulares y emplear bibliotecas de programación existentes.

La cercanía de C con el nivel de la máquina también acarrea mayor complejidad en algunas ocasiones, como a la hora de gestionar la memoria, e incluso problemas de seguridad si no se trabaja con cuidado, como el famoso y temido *buffer overflow*[3].

Este lenguaje de programación es usado, junto a las bibliotecas GLib y GObject, para el desarrollo de TempusFugit.

2.5.2. GLib y GObject

Aunque C es un buen lenguaje para programar, tiene algunas carencias: la biblioteca estándar no es tan rica como las de otros lenguajes y no es orientado a objetos. Las bibliotecas GLib [13] y GObject [14] forman parte de las herramientas de desarrollo de GNOME y surgen con la idea de paliar esas lagunas.

GLib es una bibliotecas de utilidades de propósito general que proporciona tipos de datos, macros, conversiones de tipos, utilidades para cadenas de caracteres y ficheros entre otras.

GObject hace realidad el desarrollo orientado a objetos en C, ofreciendo un sistema de tipos y otras características de los lenguajes orientados a objetos. Además está diseñado para trabajar con otros lenguajes a través de *bindings*[4].

Cabe destacar que ambas bibliotecas no son dependientes del resto del entorno de escritorio GNOME, y que están disponibles en varias plataformas, por lo que se mantiene en buena medida la independencia de la máquina en el código escrito utilizando estas bibliotecas.

El desarrollo de TempusFugit es realizado sobre la base que proporcionan C junto a estas dos bibliotecas.

[3] El error *buffer overflow* —o desbordamiento de *buffer*— ocurre cuando se copian en un área de memoria más datos de los que puede contener, existiendo la posibilidad de sobreescribir otros datos existentes.

[4] Un *binding* es una adaptación de una biblioteca de programación a otro lenguaje.

2.5.3. D-Bus

D-Bus [15] es un sistema de comunicación entre procesos para aplicaciones software, con el fin de comunicarse entre sí. Es desarrollado como parte del proyecto Freedesktop.org.

El mecanismo de comunicación entre procesos de D-Bus se divide en varias capas:

1. Una biblioteca que permite a dos aplicaciones conectarse e intercambiar mensajes.

2. Un demonio ejecutable al que pueden conectarse aplicaciones y que se encarga de entregar los mensajes que se envían las aplicaciones entre sí.

3. Bibliotecas adaptadas para su uso en varios lenguajes de programación.

Utilizando este sistema, dos aplicaciones en la sesión de un usuario pueden comunicarse entre sí e intercambiar datos variados.

Este sistema de comunicación se ha empleado para conocer el estado del salvapantallas y detectar así periodos de inactividad en la sesión del usuario.

2.5.4. XML

XML [16], siglas en inglés de Extensible Markup Language (lenguaje de marcas extensible), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).

No se trata de un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades, proponiéndose como un estándar para el intercambio de información estructurada entre diferentes plataformas.

DeTraS emplea documentos XML para almacenar la información que recopila en el ordenador del usuario de la aplicación, documentos que son tratados posteriormente para guardar esa información en una base de datos.

2.5.5. Python

Python [17, 18] es un lenguaje de programación creado por Guido van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses

«Monty Python». Se trata de un lenguaje interpretado, con *tipado* dinámico, fuertemente *tipado* y orientado a objetos. Posee una sintaxis muy limpia y que favorece un código legible.

El intérprete de Python está disponible en multitud de plataformas (UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS, etc.) por lo que si no utilizamos librerías específicas de cada plataforma nuestro programa podrá correr en todos estos sistemas sin grandes cambios. Además, todo el proyecto es gratuito y de código abierto, con todas las ventajas que ello conlleva.

Este lenguaje de programación se emplea en varios componentes de DeTraS: el *applet* para GNOME, el programa Squealer, el servicio web, la generación de informes y la aplicación web Dazer.

2.5.6. Django

Django [19] es un *framework* de código libre para el desarrollo de aplicaciones web, escrito en Python, que sigue el patrón arquitectónico Modelo–Vista–Controlador

El principal objetivo de Django es permitir la creación de sitios web complejos de forma sencilla, incentivando la reusabilidad y la conexión entre componentes.

La aplicación web Dazer ha sido construida empleando el entorno y las utilidades que Django provee.

2.5.7. HTML

HTML [20], siglas de *HyperText Markup Language* (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de «etiquetas», rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir *scripts* (por ejemplo Javascript) que pueden influir en el comportamiento de navegadores web y otros procesadores de HTML.

Para mostrar los contenidos de Dazer en un navegador web se emplea este lenguaje de marcado.

2.5.8. JavaScript y Dojo

JavaScript [21] es un lenguaje de programación débilmente *tipado*, interpretado y con capacidades de orientación a objetos. Sintácticamente, el lenguaje JavaScript recuerda a C, C++ y Java, aunque sus similitudes terminan ahí.

Lo más común es que sea empleado en navegadores web y, en ese contexto, su propósito general es permitir a los *scripts* interactuar con el usuario, controlar el navegador y alterar el documento que aparece en la ventana del mismo.

Dojo [22] es un *framework* que contiene una potente API y otras utilidades para facilitar el desarrollo de aplicaciones web, especialmente aquellas que utilizan tecnología AJAX.

Para mejorar la experiencia de usuario en la aplicación web Dazer, se ha recurrido al lenguaje JavaScript, complementado con la biblioteca que proporciona Dojo para trabajar con una mayor comodidad y ayudar a que el código producido funcionase en el mayor número de navegadores existentes.

2.5.9. CSS

Las hojas de estilo en cascada [23] (*Cascading Style Sheets* o CSS) son un lenguaje empleado para la presentación de documentos escritos en un lenguaje de marcado, es decir, nos permite modificar el formato y la apariencia del documento. Su aplicación más usual es formatear páginas web escritas en HTML y XHTML, aunque el lenguaje también puede aplicarse a cualquier tipo de documento XML.

Dentro del ámbito del presente proyecto, las hojas de estilo CSS han sido empleadas para dar la apariencia actual a la aplicación web desarrollada.

2.5.10. Google Chart API y Python Google Chart

Google Chart API [24] es una herramienta que permite generar gráficas dinámicamente a partir de parámetros que se indican en una URL. Google crea una imagen PNG a partir de los datos pasados en la petición HTTP, permitiendo al usuario descargar la imagen o mostrarla embebida en una página web usando una etiqueta de imagen.

Python Google Chart [25] (también conocido como pygooglechart), es un envoltorio para emplear Google Chart API de una manera más cómoda y escalable en Python.

Ambas tecnologías se han empleado para mostrar las gráficas producidas en Dazer.

2.5.11. SQLite

SQLite [26, 27] es una biblioteca software que implementa un sistema de gestión de bases de datos relacional. A diferencia de otros SGBD, SQLite no es un proceso independiente con el que el programa principal se comunica. En su lugar, la biblioteca SQLite se enlaza con el programa y es empleado a través de llamadas a la misma.

El conjunto de la base de datos es almacenado como un sólo fichero en la máquina donde se ejecuta el programa, reduciendo la latencia de las llamadas a la base de datos.

DeTraS emplea SQLite por defecto para gestionar la base de datos que contiene la información obtenida en los ordenadores de los clientes. No obstante, cabe destacar que el sistema está preparado para poder emplear otro SGBD si fuese necesario.

2.5.12. Glade, GTK+ y PyGtk

GTK+ [28] o *The GIMP Toolkit* es un conjunto de bibliotecas multi–plataforma para desarrollar interfaces gráficas de usuario, principalmente para los entornos gráficos GNOME, XFCE y ROX aunque también se puede usar en el escritorio de Windows, MacOS y otros. Inicialmente fueron creadas para desarrollar el programa de edición de imagen GIMP, sin embargo actualmente se usan bastante por muchos otros programas en los sistemas GNU/Linux. Junto a Qt es una de las bibliotecas más populares para X Window System.

PyGTK [29] es un *binding* en Python para la biblioteca gráfica GTK+. De este modo, es posible desarrollar interfaces gráficas potentes empleando GTK+ en Python.

Glade [30] es una herramienta de desarrollo visual de interfaces gráficas mediante GTK/GNOME. Es independiente del lenguaje de programación y genera un archivo XML que puede emplearse para construir la interfaz gráfica de usuario en tiempo de ejecución mediante bibliotecas.

Estos tres elementos son claves en el desarrollo del *applet* para el escritorio GNOME de DeTraS. Como puede adivinar, la interfaz gráfica fue diseñada con Glade, implementando su comportamiento empleando PyGTK.

2.5.13. Otras

Además de las herramientas y tecnologías detalladas con anterioridad, me gustaría destacar otras que no componen por sí mismas los componentes del proyecto, pero

que han sido claves para facilitar el desarrollo del mismo.

Bazaar

Bazaar [31] es un sistema de control de versiones distribuido patrocinado por Canonical Ltd., diseñado para facilitar la contribución en proyectos de software libre y código abierto.

Este sistema de control de versiones ha sido empleado durante todo el desarrollo del proyecto.

Launchpad

Launchpad [32] es una aplicación web para el soporte del desarrollo software, especialmente del software libre, ofreciendo las siguientes características: alojamiento de código, sistema de seguimiento de errores, sistema de seguimiento de especificaciones y nuevas características, base de conocimiento de la comunidad, facilidades para localizar aplicaciones, y una herramienta que permite construir y mantener repositorios de paquetes.

A lo largo del desarrollo del proyecto, he venido empleando varias de las herramientas existentes en Launchpad, puesto que permiten centralizar el desarrollo de DeTraS en un único lugar, que está disponible para mí y para otros miembros que forman —y formarán— parte de la comunidad creada en torno al mismo.

Herramientas para compilar y construir la aplicación

autogen y automake. Facilitan la configuración, compilación, instalación y limpieza de los componentes del proyecto.

chroot. Permite disponer de entornos de pruebas dentro del propio sistema con total seguridad. Se ha utilizado para construir y probar la generación de paquetes Debian.

dh_make y dpkg-buildpackage. Han sido empleados para la generación de paquetes Debian.

Capítulo 3

Objetivos

3.1. Propósito del proyecto

Como se ha comentado previamente, este proyecto pretende ampliar, mejorar y difundir el sistema DeTraS descrito en el apartado 2.4. Por tanto, el propósito de ambos proyectos es el mismo: obtener informes de actividad de los desarrolladores de forma automática —capturando las distintas tareas que efectúan en su ordenador—, con el fin de observar y analizar el esfuerzo dedicado a las distintas fases del desarrollo de una aplicación.

Hacer que el sistema sea autónomo conlleva numerosas ventajas respecto a los sistemas que requieren una participación activa del desarrollador:

- No interrumpe ni retrasa el desarrollo de las labores del desarrollador.
- Registra toda la información requerida, obteniendo una precisión elevada.
- La recolección de datos realizada, junto a otras herramientas disponibles para analizar diversos aspectos del desarrollo de proyectos software, permiten identificar y separar, de forma bastante acertada, las actividades correspondientes al desarrollo frente a otras tareas.

3.2. Descripción de objetivos

Es preciso describir brevemente los distintos objetivos que se pretenden conseguir al desarrollar este proyecto, partiendo de la base existente de DeTraS comentada en la sección 2.4.

3.2.1. Detección de inactividad

No es posible ofrecer datos ajustados sobre el tiempo dedicado a las distintas tareas realizadas por el usuario si no tenemos en cuenta los periodos de inactividad de la sesión que estamos observando. Para conseguir este objetivo hay que trabajar en varios sentidos: en primer lugar, controlando el periodo de tiempo durante el cual no se han empleado el teclado y el ratón, asumiendo que el ordenador no está siendo utilizado tras un plazo razonable de tiempo; por otro lado, detectando el momento en el que el salvapantallas es activado —ya sea por voluntad del usuario o porque esté programado— y desactivado; por último, es preciso tener en cuenta algunos hitos que ocurren durante el uso del ordenador, como puede ser el momento en el que se ordena el apagado del mismo, o cuando la computadora queda suspendida.

En el estado inicial de DeTraS se contempla la detección de inactividad basándose en el uso del teclado y el ratón, de igual modo que la detección de eventos relacionados con la sesión del usuario. Por tanto, un objetivo de este proyecto es conseguir que el sistema también identifique los periodos de inactividad basándose en los momentos de activación y desactivación del salvapantallas.

3.2.2. Filtrado de información

Tras leer el propósito del proyecto, una de las primeras preocupaciones que se plantean es la privacidad de los desarrolladores. Con el fin de enviar únicamente los datos deseados por el usuario de la aplicación al servidor encargado de recibir los informes de actividades, se había propuesto realizar un sistema de filtrado que nunca se llegó a desarrollar.

Es imperativo para proteger la privacidad de los futuribles usuarios del sistema DeTraS la implementación de distintos niveles de filtrado de los informes, que puedan ser elegidos por los propios usuarios, y que permitan restringir la información que es suministrada al servidor.

3.2.3. Mejoras en el *applet*

En realidad, este punto puede entenderse como un conjunto de objetivos en sí mismo, aunque todos giran en torno a ampliar las funcionalidades del *applet* desarrollado para el entorno de escritorio GNOME.

Las principales adiciones que quieren realizarse son las siguientes:

- Visualización de los datos del propio usuario.
- Posibilidad de configurar parámetros del sistema —tales como el tiempo para considerar que el equipo está inactivo, la configuración de la conexión con el servidor, o el nivel de filtrado— mediante una interfaz gráfica, proporcionada por el propio *applet*.
- Activar y desactivar la observación de las actividades desarrolladas y/o el envío de datos al servidor.

3.2.4. Visualización de resultados globales

Hasta ahora, para generar informes a partir de los datos almacenados en el servidor, era necesario emplear pequeños *scripts*. Aunque esta puede tratarse de una forma más o menos cómoda de acceder a la información una vez se disponen de los *scripts*, no cabe duda de que sería interesante crear una aplicación web que permitiese acceder a una serie de informes sobre las actividades desarrolladas y que han sido almacenadas en el servidor —por supuesto, respetando las preferencias de filtrado establecidas por los usuarios— a través de un cliente ligero, es decir, de un navegador web.

3.2.5. Facilitar la instalación de la aplicación

Para conseguir una instalación más sencilla, se crearán paquetes de software para algunas de las distribuciones de GNU/Linux más importantes en la actualidad. Cabe destacar que será necesario realizar una buena separación de los contenidos de cada paquete, haciendo posible instalar de forma independiente las distintas partes de la aplicación.

Capítulo 4

Descripción informática

Este capítulo es de vital importancia para comprender el software desarrollado a lo largo de este proyecto, así como la forma en que se ha llevado a cabo este proceso.

4.1. Metodología

El desarrollo de todo producto software se realiza bajo un determinado modelo de proceso que establece una serie de pasos a realizar, el orden en el que deben ejecutarse y sus relaciones para, partiendo del problema que se pretende resolver inicialmente, guiar el desarrollo hasta el fin de la vida del producto software implementado para resolver el problema.

En la actualidad, existen multitud de procesos software que se adaptan mejor a determinadas circunstancias que a otras. No existe un proceso que se adapte a todas las situaciones, por lo que es preciso analizar detenidamente el proyecto que se va a realizar antes de tomar una decisión.

Para desarrollar el presente proyecto, he escogido un modelo de proceso en espiral, debido a que he considerado que era el que mejor encajaba con el software existente y los objetivos marcados. Los motivos para tomar esa decisión pueden consultarse en la sección 4.1.2.

4.1.1. Modelo en espiral

Este modelo fue definido por primera vez en 1988 por Barry Boehm, y es considerado uno de los modelos de desarrollo más avanzados.

El proceso se representa como una espiral más que como una secuencia de actividad-

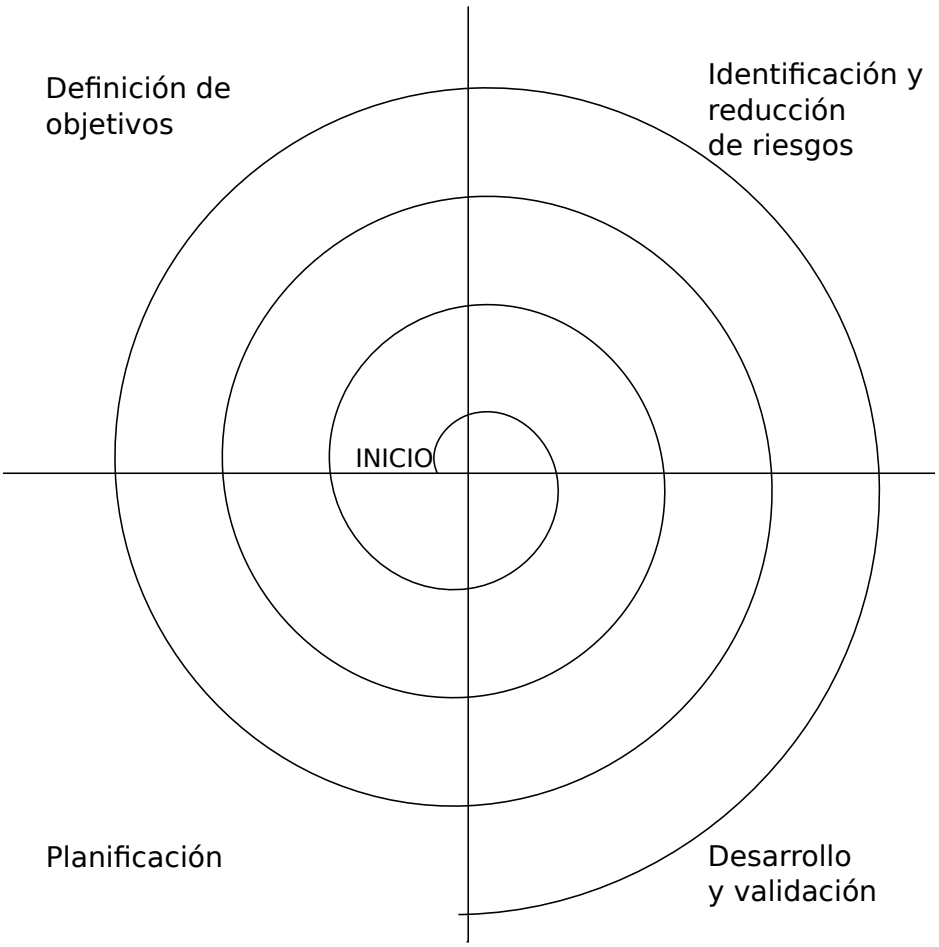

Figura 4.1: Modelo de desarrollo en espiral.

des con vuelta hacia atrás, permitiendo integrar el resultado de cada vuelta a la espiral con el resultado anterior. Además, no hay «etapas» fijas tradicionales, ligadas a actividades como la especificación o el diseño, sino que cada vuelta que se da a la espiral determina las actividades a realizar.

El desarrollo en espiral se puede dividir en las siguientes etapas:

Definición de objetivos. Se identifican los objetivos específicos para cada fase del proyecto.

Identificación y reducción de riesgos. Se identifican y analizan los riesgos clave, sirviendo esta información para minimizarlos.

Desarrollo y validación. El sistema se desarrolla y valida usando casos de prueba, comprobando que los requisitos especificados se cumplen de manera satisfactoria.

Planificación. Se revisa el proyecto y se trazan planes para la siguiente vuelta a la espiral.

4.1.2. Aplicación del modelo al proyecto

Una vez analizadas las características del modelo en espiral, puede observar que este modelo permite realizar cambios incrementalmente, es decir, cada iteración de la espiral tiene unos objetivos definidos y su conclusión integra sus resultados con el resto del sistema desarrollado hasta entonces. Si se detiene a observar los objetivos del presente proyecto, coincidirá conmigo en que son muy variados, obligando a realizar modificaciones en prácticamente todos los componentes de DeTraS, además de suponer la creación de otros componentes nuevos. Este tipo de modificaciones y ampliaciones se adaptan muy bien a los cambios incrementales que se producen en el modelo en espiral, permitiendo culminar uno o varios de ellos en cada iteración.

Generalmente, el proceso de desarrollo del software debe contar con la supervisión e indicaciones del cliente, que debe asegurarse que el producto que estamos realizando se adapte a sus necesidades. En el caso que nos concierne, no existe un cliente como tal, sino que el cliente es cualquier usuario potencial de la aplicación. Por ello, sería interesante la posibilidad de lanzar versiones incrementales que permitan obtener realimentación por parte de los posibles usuarios de la aplicación. Nuevamente, el modelo

en espiral se ajusta a esa situación, ya que es posible, una vez el desarrollo del producto sea lo suficientemente maduro, obtener al final de cada iteración una versión del software cada vez más completa. Considero que la versión inicial de DeTraS, tras una primera iteración que permita detectar y corregir algunos fallos existentes, puede ser un buen punto de partida para emplear este recurso que ofrece el modelo en espiral.

Además de las dos razones explicadas con anterioridad, una de las principales ventajas del modelo de desarrollo en espiral es su facilidad para agregar nuevos requisitos. Esto puede resultar de especial utilidad para corregir posibles errores que se detecten durante el desarrollo del proyecto, o para permitir modificar el plan trazado si se desease alterar algún requisito del software.

4.2. Arquitectura general

El sistema DeTraS está formado por un conjunto de herramientas que permiten seguir la actividad de desarrolladores. Si se analiza este sistema como un todo, se puede advertir que sigue un modelo de arquitectura cliente–servidor.

Las arquitecturas cliente–servidor están integradas por tres componentes: el cliente, el servidor y la forma en que se comunican. En el caso de DeTraS, el cliente será cada uno de los desarrolladores que utilicen la aplicación; el servidor será una máquina dedicada que recibirá los datos de cada uno de los clientes y servirá informes e información a partir de los datos recopilados; y ambos componentes se conectarán a través de una red de ordenadores empleando los protocolos HTTP y HTTPS.

La figura 4.2 muestra gráficamente la arquitectura descrita en las líneas anteriores.

4.2.1. Cliente

El cliente es el componente del sistema que se ejecuta en el ordenador de cada uno de los usuarios, existiendo, por tanto, un cliente por cada usuario que tenga la aplicación.

En el sistema DeTraS, la parte cliente está compuesta por dos aplicaciones:

Observador. Es el encargado de detectar y registrar las actividades realizadas por el usuario de la aplicación. El nombre de este componente de DeTraS es TempusFugit.

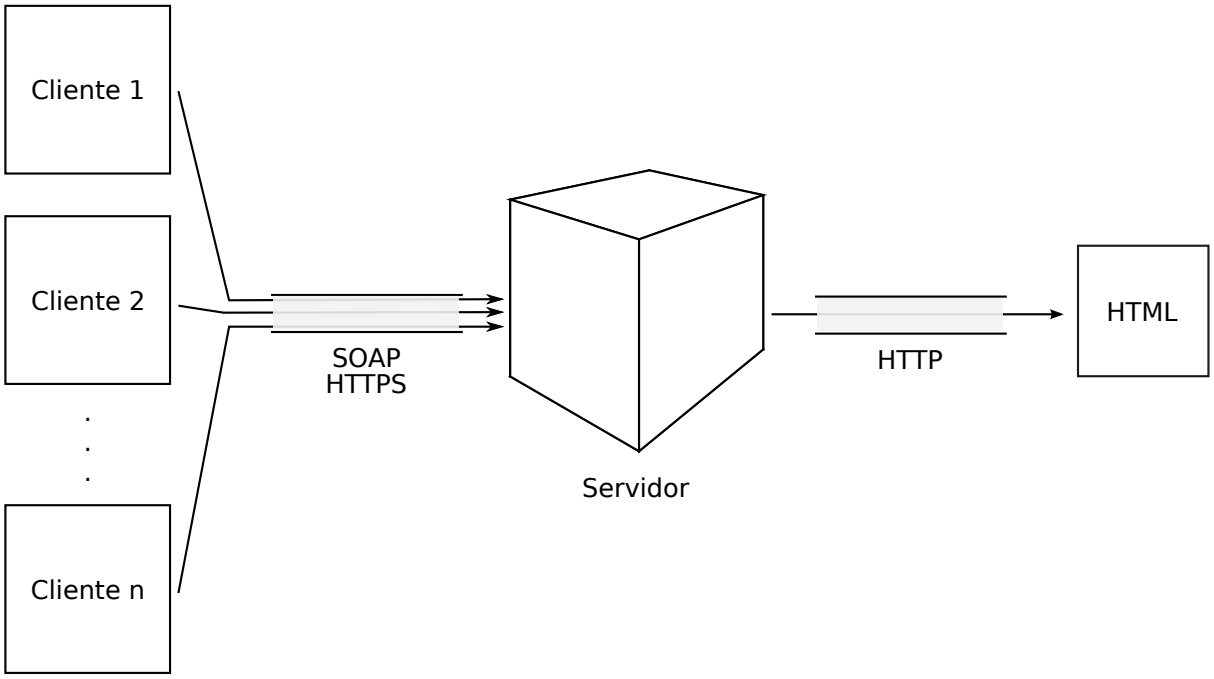

Figura 4.2: Diagrama de la arquitectura de DeTraS.

Notificador. Su nombre es Squealer y tiene como misión enviar los datos recogidos por TempusFugit al servidor de la aplicación.

Cabe resaltar que estos dos programas disponen de un conjunto de herramientas a su alrededor que complementan y facilitan el uso de las mismas.

4.2.2. Servidor

Este elemento del sistema ejecutará habitualmente en una máquina remota, donde se recibirán y almacenarán de forma persistente los datos enviados por los clientes. Asimismo, ofrecerá la posibilidad de obtener informes e información útil a partir de los datos recogidos.

El servidor de DeTraS está compuesto por las siguientes herramientas:

Recogida de datos. Servicio web cuya tarea es realizar la recepción y almacenamiento de los datos que envían los distintos clientes de DeTraS.

Generador de informes. Accediendo a la base de datos permite realizar informes sobre la actividad de los desarrolladores.

Analizador de datos. Esta aplicación web, llamada Dazer, facilita la generación de gráficas y la obtención de información útil partiendo de los datos almacenados en el servidor. Los clientes de esta aplicación serán aquellos usuarios que accedan a la misma a través de su navegador web.

4.2.3. Comunicación

En el caso de los clientes que envían sus datos al servidor empleando Squealer, la comunicación se llevará a cabo mediante SOAP sobre HTTPS, permitiendo el intercambio seguro de datos entre las aplicaciones, evitando que sean filtradas por cortafuegos o que sean leídas por un atacante.

Cuando los clientes accedan a la aplicación web Dazer, se comunicarán con la misma a través del protocolo HTTP, empleando para ello el navegador web de su preferencia.

4.3. Iteración inicial. Estudio de la aplicación

4.3.1. Especificación

Puesto que el presente proyecto se construye sobre la base dejada por Carlos García Campos, es indispensable conocer con un buen nivel de detalle las distintas partes desarrolladas en su momento, analizar y comprender su estructura interna, así como verificar su funcionamiento.

A la hora de desarrollar software, es imprescindible conocer si se cumplen los requisitos marcados en un principio y si el software resulta de utilidad. Esto puede resultar relativamente sencillo de estimar cuando realizamos un software por «encargo», por ejemplo, entrevistando a los usuarios de la aplicación. No obstante, al realizar un proyecto software sin un cliente concreto, la recogida de esa información se complica bastante, resultando interesante conocer la situación en la que se encuentra el proyecto antes de comenzar a trabajar, para tener unas determinadas expectativas sobre su futuro. Por ejemplo, puede ser interesante analizar el número de *commits* efectuados durante el último año para deducir si el desarrollo del proyecto se encuentra activo o no. Realizar una estimación del número de usuarios de la aplicación también puede ayudar a valorar la colaboración y realimentación que se obtendrá por parte de los mismos.

Resultado

El resultado obtenido en esta iteración ha sido una comprensión mucho mayor de la estructura y estado en que se encontraba el software. Cabe destacar que esta etapa ha sido dura, puesto que continuar un producto software existente, como sabrá todo aquel que se haya enfrentado a este problema, requiere de una buena comprensión de las tecnologías empleadas en el proyecto, la realización de una ardua tarea de documentación en diversos aspectos, efectuar un estudio a fondo del código existente, así como completar una serie de pruebas. Todo ello para entender los motivos por los que se llevaron a cabo las decisiones de diseño tomadas, determinar si estas decisiones eran correctas y conocer el estado en que se encuentra el proyecto.

La versión del código disponible cuando se comenzó el presente proyecto constaba de TempusFugit, Squealer, el servicio web de recogida de datos y la aplicación para generar informes, ascendiendo la suma de sus líneas de código a unas cinco mil.

A la hora de analizar el estado del proyecto en lo que respecta a su actividad, el último *commit* realizado data de mayo de 2007, por lo que se puede concluir que el desarrollo estaba parado en el momento de comenzar el presente proyecto. No he podido encontrar ningún dato que indique un número de usuarios —aunque fuese aproximado— de la aplicación, así que, viendo el tiempo que llevaba inactivo el proyecto, considero que el número de usuarios sería más bien escaso.

4.4. Primera iteración

4.4.1. Especificación

Una vez se conoce en detalle cómo se encuentra el desarrollo de la aplicación, su estructura y se han realizado un buen número de pruebas con las distintas partes que la componen, es el momento de arreglar los problemas encontrados para obtener una versión del código robusta sobre la que se realizarán el resto de modificaciones.

En concreto, se han encontrado los siguientes problemas:

- Existencia de un error a la hora de liberar memoria en la clase `EventWc`.
- Si justo después de ordenar la detención de TempusFugit se cambia de ventana, se apunta ese evento después del que indica el fin de sesión.

- Errores en el *applet* de GNOME, cuya solución se pospondrá a próximas iteraciones, cuando se trabajará en su desarrollo.

Además, se aprovechará esta iteración para incluir y probar los datos del servidor que el Grupo de Sistemas y Comunicaciones —también conocido como GSyC— ha prestado generosamente para albergar el servidor de este proyecto.

4.4.2. Diseño

Para evitar que TempusFugit siga agregando eventos una vez se ha ordenado su cierre, es necesario desconectar los *listeners* que escuchan los eventos en la clase `Observer` de modo que, aunque se produzcan estos eventos posteriormente, no se efectúe ninguna acción.

4.4.3. Implementación

El error en la liberación de memoria en la clase `EventWc` se ha solucionado agregando la llamada pertinente a `free` en el lugar donde era necesaria.

Sobre el error al detener TempusFugit, se ha solucionado almacenando los identificadores de los manejadores de los eventos que se generan al cambiar de ventana o entrar en un periodo de inactividad, de manera que es posible desconectarlos posteriormente, evitando que se reciban más eventos cuando se va a finalizar la ejecución de TempusFugit.

4.5. Segunda iteración

4.5.1. Especificación

A continuación se detallan las distintas tareas que se han llevado a cabo en esta iteración:

Detección del estado del salvapantallas

Si se desea realizar una herramienta que realice un seguimiento sobre el tiempo de uso de un ordenador, es imprescindible detectar correctamente los periodos de inactividad de los usuarios. Uno de los pilares para conseguirlo es observar el comporta-

miento del salvapantallas, teniendo en cuenta que el usuario estará inactivo cuando el salvapantallas esté funcionando.

Mejoras en el manejo de errores de Squealer

Hasta ahora, cuando ocurría un error en la ejecución de Squealer, se mostraba por pantalla toda la traza del error, dando lugar a unos mensajes difícilmente comprensibles por un usuario de la aplicación. Para paliar esta situación, en esta iteración se tratarán los errores intentando producir mensajes más simples y descriptivos.

Resolución de errores

Aunque la iteración anterior tuvo como principal objetivo la resolución de errores, se han detectado con posterioridad otros dos que se corregirán a lo largo de esta iteración, y que se mencionan a continuación:

1. Al iniciar TempusFugit no se registraba la primera ventana que tuviese el foco, por lo que si el usuario estaba trabajando en ella esa información se perdía.

2. Cuando el fichero donde se almacenan los eventos existía pero estaba vacío TempusFugit generaba un fichero XML no válido.

Ayudas a la depuración

Con el fin de permitir depurar de una forma más cómoda la aplicación TempusFugit, se va a trabajar en dos sentidos:

1. Creación de un modo de depuración —o *debug*— que dé la posibilidad al usuario de mostrar datos de depuración por pantalla.

2. Desarrollo de la opción *file*, que permita al usuario indicar un destino diferente para el fichero de eventos.

4.5.2. Diseño

Detección del estado del salvapantallas

En GNU/Linux, al contrario que en otras plataformas, existen varios programas que actúan como salvapantallas. En principio, he decidido que únicamente se dará

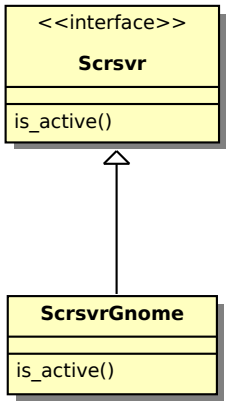

Figura 4.3: Jerarquía de clases para monitorizar el salvapantallas.

soporte a GNOME Screensaver, el salvapantallas usado por defecto en GNOME y que es empleado por una buena cantidad de distribuciones en la actualidad. No obstante, aunque no se soporten más salvapantallas por defecto, se ha tenido en cuenta a la hora de diseñar el sistema la posibilidad de agregar otros salvapantallas en el futuro.

Para llevar a cabo esta funcionalidad, se ha pensado en implementar una jerarquía de clases como la mostrada en la figura 4.3. Así, si en el futuro se desea agregar soporte para otro salvapantallas, simplemente habría que crear la clase correspondiente que implemente la interfaz `ScrsvrIface`.

Mejoras en el manejo de errores de Squealer

Los cambios necesarios para completar esta funcionalidad se efectuarán sobre el módulo `main` de Squealer, afectando a los métodos `print_error` y `main`.

Resolución de errores

En primer lugar, se detalla cómo se resolverá el problema que impide a la primera ventana ser registrada. Este error se debe a que el método que registra los cambios de aplicación únicamente es llamado cuando se detecta un cambio de foco. Como al ejecutar TempusFugit no se realiza ese cambio de foco, es preciso ejecutar manualmente el método para que anote la aplicación que lo tiene.

En lo que respecta al error con el formato del fichero de eventos, se va a agregar la lógica necesaria en la clase `RegisterXml` para que ese fichero se inicialice con la estructura necesaria. Además, para evitar que el fichero de eventos esté mal formado,

se va a crear una nueva clase, llamada `XmlValidator`, que se encargará de comprobar la estructura del fichero al comenzar la ejecución del programa.

Ayudas a la depuración

Para desarrollar el modo de depuración, se modificará la clase `Application`, añadiendo el método `debug` a la misma. De este modo, al ser `Application` una clase estática, cualquier clase de TempusFugit podrá imprimir sus mensajes de depuración llamando al método `debug`.

En el caso de la opción *file*, será preciso hacer que la clase `RegisterXml` pueda almacenar el XML generado en un fichero que se le indique. En caso de no indicarle un fichero en concreto, lo almacenará en uno por defecto, que coincidirá con el usado actualmente.

4.5.3. Implementación

Detección del estado del salvapantallas

A la hora de llevar a la práctica el diseño especificado en el apartado 4.5.2, se han implementado las clases correspondientes a la jerarquía indicada.

Para comunicarse con GNOME Screensaver se ha empleado el sistema D-Bus, del que se puede leer más en la sección 2.5.3. Realizar este proceso usando GLib es bastante sencillo, y se ha llevado a cabo siguiendo los siguientes pasos, que pueden verse reflejados en el listado 4.1:

1. Se realiza la conexión con D-Bus a través de la llamada `dbus_g_bus_get`.

2. Se crea un nuevo objeto *proxy* para GNOME Screensaver. Este objeto permitirá realizar la comunicación con el otro proceso.

3. Cuando se desee llamar al método remoto se debe utilizar la llamada a `dbus_g_proxy_call`, indicando el nombre del método que se desea ejecutar.

4. Finalmente, cuando se desee finalizar la comunicación es preciso eliminar el objeto *proxy* con `g_object_unref` y cerrar la conexión con D-Bus usando `dbus_g_connection_unref`.

```
1  static void
2  tf_scrsvr_gnome_init (TfScrsvrGnome *ssg)
3  {
4      (...)
5      ssg->priv->connection = dbus_g_bus_get (DBUS_BUS_SESSION, &error)
           ;
6      if (error) {
7          g_warning ("Failed to connect to bus: %s", error->message);
8          g_error_free (error);
9          error = NULL;
10     }
11
12     ssg->priv->proxy = dbus_g_proxy_new_for_name (ssg->priv->
           connection,
13                GNOME_SCRSVR_SERVICE, GNOME_SCRSVR_PATH,
                      GNOME_SCRSVR_IFACE);
14 }
15
16 static gboolean
17 tf_scrsvr_gnome_is_active (TfScrsvr *ss)
18 {
19     (...)
20     if (!ssg->priv->proxy)
21         return FALSE;
22
23     if (!dbus_g_proxy_call (ssg->priv->proxy, "GetActive", &error,
24                G_TYPE_INVALID, G_TYPE_BOOLEAN, &response, G_TYPE_INVALID
                      )) {
25         g_error_free (error);
26         return FALSE;
27     }
28
29     return response;
```

Listado 4.1: Código para comunicarse con otro proceso usando D-Bus.

Finalmente, es necesario agregar la comprobación del estado del salvapantallas en la clase `Idle`, de modo que se cree un evento de inactividad cuando el salvapantallas esté activo.

Mejoras en el manejo de errores de Squealer

El primer cambio realizado consiste en imprimir un mensaje de error diferente dependiendo del lugar del código donde se produzca la excepción. De este modo el usuario conocerá el proceso que estaba realizando el programa en el momento del fallo.

Además de esa modificación, se ha cambiado el método empleado para imprimir el mensaje de error, usando uno que produce menos cantidad de información pero es más inteligible para un usuario común.

Resolución de errores

Para solventar el problema detectado con el registro de la primera aplicación usada, basta con realizar una llamada al método `tf_observer_active_window_changed` en el constructor de la clase `Observer`.

La resolución del fallo en la estructura del documento XML que almacena los eventos es algo más compleja. Por un lado, se ha implementado el método `tf_register_xml_init_file`, que inicializa un fichero de texto para que contenga la estructura requerida por el documento XML que registra los eventos. Este método es llamado por el método `constructed` de la clase `RegisterXml` en caso que el fichero de eventos exista y esté vacío. Si el fichero de eventos, por el contrario, contiene datos, se llamará a la clase `XmlValidator` —creada en esta iteración— para que compruebe su estructura.

En lo que respecta a `XmlValidator`, no validará el documento de eventos frente a ningún esquema XML, ya que su formato es ambiguo y no puede representarse con un esquema. No obstante, esta clase comprobará la estructura básica del documento —esencialmente, asegurándose que comienza y concluye con las etiquetas correctas—, evitando la modificación de ficheros erróneos generados con versiones anteriores de TempusFugit. Asimismo, el diseño de esta clase deja la puerta abierta a una validación más exhaustiva si un rediseño del fichero que almacena los eventos lo permite.

Ayudas a la depuración

El modo de depuración se ha desarrollado siguiendo los pasos descritos en el apartado de diseño, es decir, implementando un método `debug` en la clase `Application`. Este método comprueba si se desea mostrar la información de depuración —confirmando que se ha utilizado el argumento que activa este modo—, e imprime por pantalla los datos indicados en su llamada si fuese preciso.

En la implementación de la opción *file* ha sido necesario, en primer lugar, agregar la opción mencionada previamente a la clase `Application`. El argumento que recibe esta opción —que corresponderá con el nombre del fichero en el que se almacenarán los datos— se pasa como parámetro al constructor de la clase `Observer`, que a su vez se lo pasará al constructor de la clase `RegisterXml`. Esta clase, en su método `constructed`, comprobará si se ha indicado algún fichero en concreto o utilizará el indicado por defecto para almacenar el texto XML generado.

4.6. Tercera iteración

4.6.1. Especificación

Filtrado de información

Es de vital importancia para el desarrollo de DeTraS permitir a los usuarios que mantengan su privacidad. Para conseguirlo, se ha pensado desarrollar una serie de filtros que eviten que el usuario pueda escoger la información que sea subida al servidor.

Tras analizar los tipos de problemas de privacidad que pueden darse empleando DeTraS, se ha concluido que hay dos focos conflictivos: los títulos de las ventanas y el envío del nombre de usuario junto a los datos del mismo.

Cambios en la configuración de DeTraS

Hasta ahora, la configuración de DeTraS se realizaba antes de efectuar la compilación. Como notará el lector, este sistema es demasiado farragoso para permitir al usuario realizar modificaciones de una forma sencilla.

Por tanto, se va a proceder a modificar el sistema de configuración —en principio de Squealer, que es el programa más *parametrizable* de los que componen DeTraS—, de manera que exista un fichero de configuración por cada usuario que contendrá sus

preferencias personales. Además, también existirá un fichero de preferencias globales para todo el sistema, que serán empleadas si el usuario aún no ha establecido las suyas.

Configuración de parámetros de la aplicación desde el *applet*

Este punto está relacionado con el anterior, y es que para hacer más sencilla la configuración de DeTraS, se pretende realizar una interfaz gráfica de usuario que permita modificar las opciones desde el *applet* de GNOME.

De hecho, debido a los problemas encontrados con el *applet* que se especifican en el apartado 4.4, este punto servirá para comenzar a construir el nuevo *applet* de DeTraS.

4.6.2. Diseño

Filtrado de información

Como mencioné en la especificación de esta iteración, existen dos focos de posibles problemas de privacidad, que se resolverán como se explica a continuación:

Títulos de las ventanas. Este problema se resolverá desarrollando un sistema que permita al usuario —además de mantener sus datos de forma íntegra— omitir partes de los títulos de las ventanas u omitir todos los títulos de las ventanas. Para ello, se creará la jerarquía de clases mostrada en la figura 4.4 que hará posible filtrar en los títulos de todas las ventanas empleando una expresión regular[1] —clase `RegExpFilter`—, filtrar todos los títulos de las ventanas —clase `WindowsTitlesFilter`— o dejar los resultados sin filtrar —clase `NoFilter`—.

Envío del nombre de usuario. Este problema se solucionará modificando la clase `Client`, haciendo que se respete la opción escogida por el usuario al respecto.

Todos estos cambios se llevarán a cabo sobre Squealer, que es el encargado de enviar los datos recogidos en el sistema del usuario al servidor. Por supuesto, para que sean efectivos deberán realizarse antes de completar el envío de los datos.

Finalmente, quería añadir que, pese a que los datos enviados al servidor estarán sujetos a los filtrados configurados por el cliente, no se modificarán los datos localmente, permitiendo que el usuario tenga la opción de consultar sus registros de eventos de forma íntegra.

[1] Una expresión regular, a menudo llamada también patrón, es una expresión que describe un conjunto de cadenas sin enumerar sus elementos.

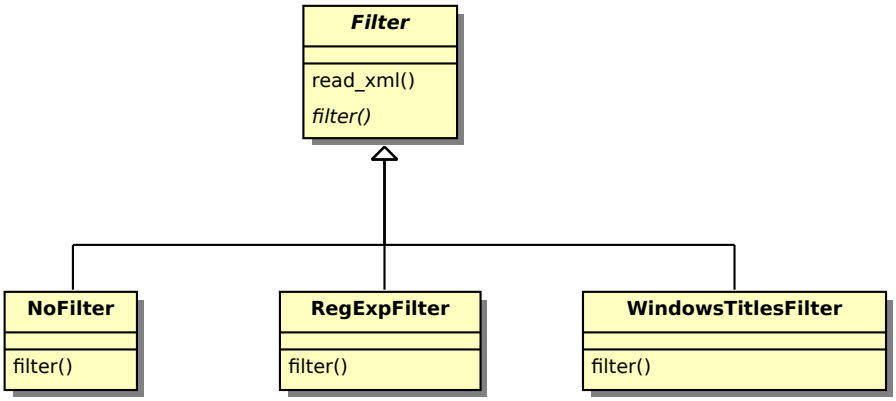

Figura 4.4: Jerarquía de clases para filtrar los datos enviados al servidor.

Cambios en la configuración de DeTraS

Para llevar a cabo este cambio es necesario, por un lado, definir la forma de almacenar las preferencias y, por otro lado, definir la forma en que se trabajará con ellas.

Para el almacenamiento de las preferencias he decidido usar un archivo INI. Se ha escogido este tipo de ficheros porque su estructura es clara y sencilla. Básicamente, el fichero consta de una serie de secciones, dentro de las cuales se definen variables con sus respectivos valores. En el listado 4.2 se puede observar la estructura de uno de estos ficheros.

Respecto a la forma de manejar el fichero de preferencias, se va a emplear una adaptación de la clase `SettingsManager`, proveniente del proyecto de software libre Exaile[2]. A su vez, con el objeto de que Squealer sea capaz de emplear esta configuración, será preciso modificar la clase `GlobalConfig` para que varíe la fuente de la que toma los datos de configuración.

Configuración de parámetros de la aplicación desde el *applet*

Como mencioné con anterioridad, la realización de este objetivo va encadenada a la creación de un nuevo *applet*, escrito en Python, sobre el que se irán añadiendo mejoras y nuevas funcionalidades. De momento, el resultado de esta iteración debe ofrecer un *applet* muy sencillo, básicamente para mostrar la ventana de configuración.

Para el diseño de la interfaz de usuario se empleará el programa Glade, que permi-

[2]Exaile es un reproductor de audio libre para el sistema operativo GNU/Linux. Dirección del proyecto: `http://www.exaile.org/`

tirá obtener una interfaz intuitiva, simple e independiente del lenguaje empleado.

Antes de desarrollar la interfaz, es necesario crear un boceto de la misma —usando papel y lápiz, o alguna herramienta diseñada para la tarea—. Estos bocetos permiten evolucionar mejor los elementos que compondrán la interfaz, así como su disposición y ordenación. En la figura 4.5 se muestran los primeros bocetos realizados para desarrollar la ventana de preferencias. Tenga en cuenta que esos bocetos han sufrido varios cambios hasta dar lugar a la apariencia de la ventana existente en la actualidad.

4.6.3. Implementación

Filtrado de información

En esta etapa se ha desarrollado la jerarquía de clases indicada en la figura 4.4. Para realizar el filtrado, las clases `RegExpFilter` y `WindowsTitlesFilter` recorren todo el documento XML buscando los elementos con la etiqueta *title*, que son los que contienen el título de la ventana. En el caso del filtro por expresiones regulares, se buscará la misma en cada título de ventana y se eliminarán todas sus apariciones. El filtro de títulos de ventanas directamente eliminará esos nodos, devolviendo un XML que no contendrá ningún título de ventana. Como puede suponer, la clase `NoFilter` no realizará ningún filtrado sobre la información, por lo que devolverá el documento XML completo.

Para aplicar estos filtros a la información del usuario, el método *main* de Squealer deberá instanciar una de las clases anteriormente mencionadas en función de las preferencias del usuario en materia de privacidad.

Una vez resuelto el problema de filtrar los títulos de las ventanas recogidos, es hora de centrarse en omitir el nombre del usuario de los datos enviados si este así lo desea. Para ello, se modificará la clase `Client`, agregando una condición que compruebe la preferencia del usuario para este asunto y genere el documento XML con su nombre en función de la misma.

Cambios en la configuración de DeTraS

Como se comentó anteriormente, se ha escogido almacenar las preferencias del usuario en un fichero INI, como el que puede observar en el listado 4.2. Las preferencias que se pueden configurar desde ese fichero se han agrupado en categorías, con

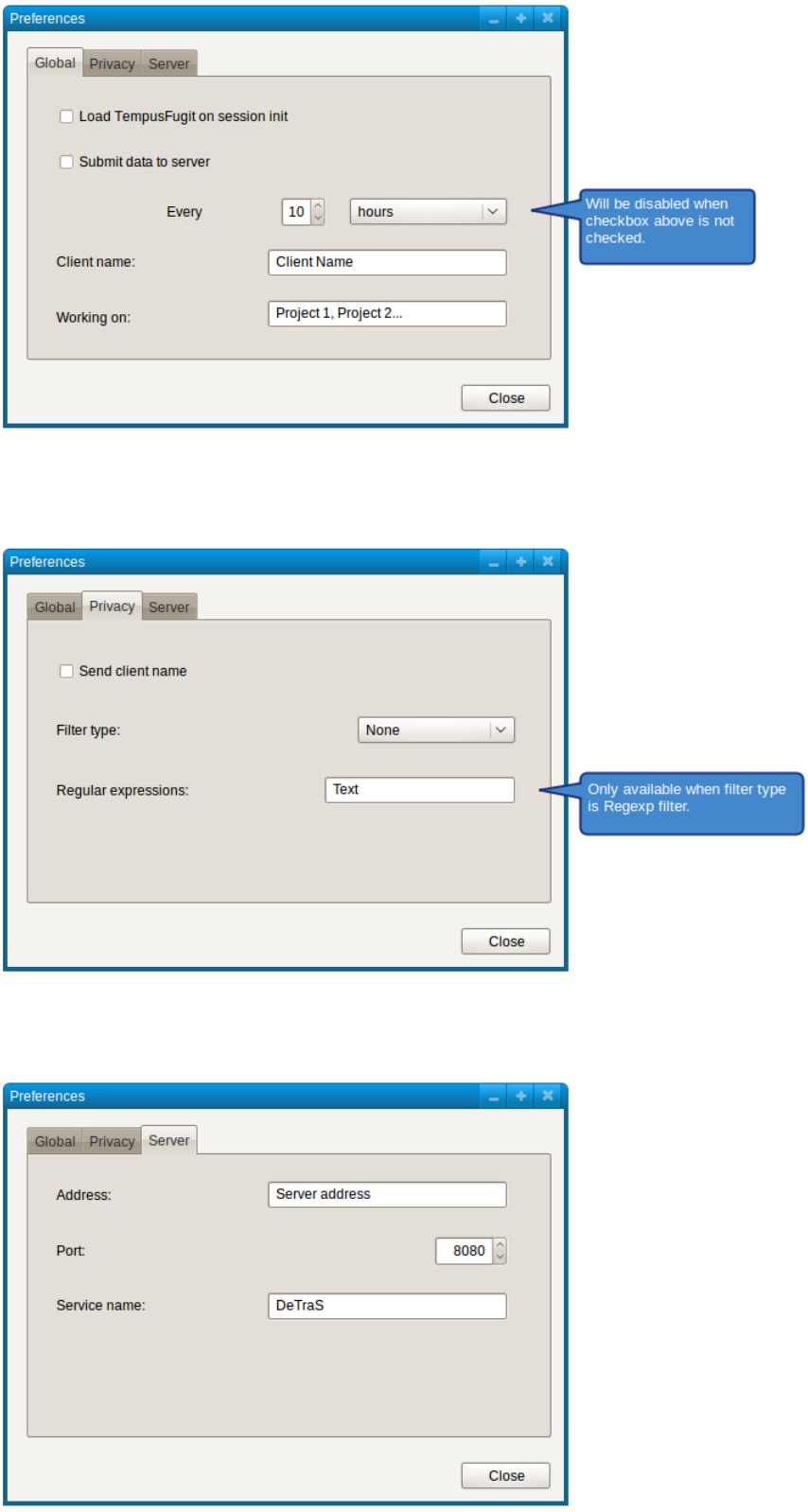

Figura 4.5: Bocetos iniciales de la interfaz de usuario de las preferencias.

el fin de permitir una edición de manera más intuitiva. Las categorías incluidas y sus preferencias se indican a continuación:

General. Configura parámetros generales de la aplicación.

Cliente. Incluye datos relativos al cliente, como su nombre y los proyectos en los que trabaja.

Servidor. Permite configurar los parámetros de conexión del servidor. Por defecto, estos parámetros están configurados con los datos de una máquina que el GSyC ha facilitado para alojar el servidor usado en este proyecto y que recoge datos para su posterior análisis.

Privacidad. Ajusta las preferencias de privacidad del usuario, permitiendo indicar si desea enviar su nombre al servidor y/o aplicar un filtro sobre los datos del cliente.

Para leer y escribir las opciones configurables en el fichero de preferencias desde la aplicación se emplea, como se anticipó, la clase `SettingsManager` que, entre otras funcionalidades, provee los métodos `get_option` y `set_option` que permiten consultar y modificar una preferencia respectivamente.

Configuración de parámetros de la aplicación desde el *applet*

En primer lugar, es preciso comenzar a construir el nuevo *applet*. Hay que tener en cuenta que los *applets* de GNOME tienen dos partes bien diferenciadas: el fichero *server* que describe las características del programa a Bonobo[3], y el código del *applet*, que proporciona funcionalidad al mismo.

Dentro del fichero *server* hay que indicar la clase a la que pertenecerá el *applet*, así como una fábrica para objetos de esa clase. Estas dos clases deberán implementarse en el código del *applet* para su correcto funcionamiento. Para aclarar al lector la estructura del fichero *server* y su funcionamiento, se incluye el contenido del creado en esta iteración —disponible en el listado 4.3—. Obsérvese que la clase `DetrasAppletFactory` representa a la fábrica de objetos `DetrasApplet`.

[3]Bonobo es un sistema de componentes basado en CORBA empleado por la plataforma GNOME para realizar aplicaciones.

```
[privacy]
; Patterns to hide from windows titles. Patterns will apply in order, so
; previous patterns may affect matching of subsequent patterns.
; E.g. ['e\.alvarezj'] will remove the string "e.alvarezj" from any window
; title.
re_filter_patterns = L: []

; Your preferred filter type.
;       0: No filter.
;       1: Filter windows titles using regular expressions (indicated in
;          re_filter_patterns).
;       2: Filter all windows titles.
filter_type = I: 0

; Set whether you want to send your name along with data tracked.
; True: you want to send your name to DeTraS server.
; False: you do not want to send your name.
send_client_name = B: False

; Data configuration for DeTraS server.
[server]
service_name = S: DeTraS
port = I: 8080
address = S: https://193.147.51.204

[client]
; Projects in which user is working on (optional).
; E.g. ['DeTraS', 'World domination']
projects = L: []

; User name (optional).
name = S:

[general]
; Filename where events are going to be saved.
events_file = S: tempusfugit.xml
```

Listado 4.2: Esqueleto del fichero de configuración.

```
<oaf_info>
  <oaf_server iid="OAFIID:DetrasAppletFactory"
      type="exe"
      location="@LIBEXECDIR@/detras-applet">
    <oaf_attribute name="repo_ids" type="stringv">
      <item value="IDL:Bonobo/GenericFactory:1.0"/>
      <item value="IDL:Bonobo/Unknown:1.0"/>
    </oaf_attribute>
    <oaf_attribute name="name" type="string" value="DeTraS␣applet"/>
    <oaf_attribute name="description" type="string" value="DeTraS␣
       configuration␣and␣control␣utility"/>
  </oaf_server>
  <oaf_server iid="OAFIID:DetrasApplet"
      type="factory"
      location="OAFIID:DetrasAppletFactory">
    <oaf_attribute name="repo_ids" type="stringv">
      <item value="IDL:GNOME/Vertigo/PanelAppletShell:1.0"/>
      <item value="IDL:Bonobo/Control:1.0"/>
      <item value="IDL:Bonobo/Unknown:1.0"/>
    </oaf_attribute>
    <oaf_attribute name="name" type="string" value="DeTraS␣applet"/>
    <oaf_attribute name="description" type="string" value="DeTraS␣
       configuration␣and␣control␣utility"/>
    <oaf_attribute name="panel:category" type="string" value="
       Accessories" />
    <oaf_attribute name="panel:icon" type="string" value="
       stock_weather-storm"/>
  </oaf_server>
</oaf_info>
```

Listado 4.3: Fichero *server* de la presente iteración.

```
<Root>
  <popups>
    <popup name="button3">
      <menuitem name="Preferences" verb="preferences" _label="
         _Preferences" pixtype="stock" pixname="gtk-preferences" />
      <menuitem name="About" verb="about" _label="_About" pixtype="
         stock" pixname="gtk-about" />
    </popup>
  </popups>
</Root>
```

Listado 4.4: XML que define el menú contextual del *applet*.

En lo que respecta al código del *applet*, primero se definió un menú contextual que permite al usuario ejecutar acciones desde el mismo. Este menú se puede definir a través de un documento XML como el que se muestra en el listado 4.4, que es el empleado en esta primera versión del *applet*. Cada elemento *menuitem* define una entrada en el menú contextual, siendo sus atributos más importantes la etiqueta que se mostrará para esa entrada —cuyo nombre es *label*— y el atributo que indica la acción que se ejecutará —llamado *verb*—.

Seguidamente, se incluyó el comportamiento de la clase `DetrasApplet`. Como puede verse en el listado 4.5, es necesario emplear el método `setup_menu_from_file` para indicar que se debe usar el fichero con la definición del menú contextual anteriormente mencionada, así como definir funciones que serán llamadas cuando se pulse sobre las distintas entradas definidas para el menú contextual.

Con el objetivo de permitir efectuar la configuración desde el propio *applet*, se ha creado la clase `PrefsDialog`, que se encarga de emplear a `SettingsManager` para rellenar los datos que se mostrarán en la interfaz diseñada con Glade cuando se cargue el diálogo y para modificar los datos que desee el usuario.

4.7. Cuarta iteración

4.7.1. Especificación

Esta iteración cuenta con varios objetivos relacionados con el desarrollo del *applet* que se comenzó a gestar en la iteración anterior.

```
1  class DetrasApplet (object):
2      def __init__ (self, applet):
3          self.applet = applet
4
5          self.applet.about = None
6          self.applet.preferences = None
7
8          self.applet.setup_menu_from_file (os.path.join (config.
               DATA_DIR, \
9                   "detras-applet"), \
10                  "DetrasApplet.xml", \
11                  None, \
12                  [("about", self.about_cb), \
13                  ("preferences", self.prefs_cb)])
14
15     def about_cb (self, component, verb):
16         if self.applet.about:
17             self.applet.about.present ()
18         else:
19             about = AboutDialog ()
20             about.show_about (self.applet)
21
22     def prefs_cb (self, component, verb):
23         if self.applet.preferences:
24             self.applet.preferences.present ()
25         else:
26             preferences = PrefsDialog ()
27             preferences.show_prefs (self.applet)
```

Listado 4.5: Clase `DetrasApplet`.

Control de TempusFugit y Squealer desde el *applet*

El *applet* desarrollado en la tercera iteración no contaba con la posibilidad de gestionar la ejecución de TempusFugit y Squealer desde el mismo, por lo que se aprovechará esta iteración para implementar esta funcionalidad.

Visualización de resultados locales

Otro de los objetivos radica en ofrecer al usuario la posibilidad de consultar, a través de una interfaz gráfica conectada al propio *applet*, los últimos eventos que han sido recogidos por la aplicación.

Mejoras en la configuración de DeTraS

La forma en que se manejaba la configuración en la iteración anterior de DeTraS era bastante mejorable, ya que tenía problemas de duplicación de código, escalabilidad y potenciales problemas de concurrencia. Por tanto, esta iteración tiene como objetivo mejorar la forma en que se gestiona la configuración de usuario en DeTraS.

Ampliación de las opciones de configuración

Además de las opciones de configuración brindadas en la iteración anterior, se desea que el usuario pueda configurar dos parámetros más de la aplicación: iniciar TempusFugit al cargar el *applet* y detenerlo cuando se elimine el *applet* del panel. Estas opciones además permiten que, cuando el usuario inicie o finalice sesión en su entorno de escritorio GNOME teniendo el *applet* agregado a un panel, TempusFugit comience a funcionar al inicio de la sesión y se detenga cuando se cierra la misma —por supuesto, únicamente si el usuario lo desea—.

Mejoras en la documentación

Pese a no ser tareas propias de programación, también es objetivo de esta iteración mejorar la documentación existente. Para ello, se va a trabajar en dos sentidos: por un lado, se crearán varias entradas de manual que permitan consultar de forma breve cómo se emplean los comandos relacionados con DeTraS —los comandos `tempusfugit` y `squealer`— y cómo modificar a mano el fichero de configuración; por otro lado, se creará una primera versión del manual de usuario de la parte cliente de DeTraS.

Facilitar la instalación de la aplicación

Durante esta iteración, también se realizarán avances en el modo de instalar la aplicación, buscando facilitar el proceso. Como recordará, este es uno de los objetivos marcados en el presente proyecto. Para alcanzar esa meta se crearán dos paquetes Debian, válidos para las distribuciones Debian y sus derivadas —entre ellas Ubuntu—, que incluirán la parte cliente de DeTraS compilada para arquitecturas Intel de 32 bits —conocida en el mundo GNU/Linux como *x86*— y de 64 bits —conocida como *amd64*—. Asimismo, se empleará la facilidad que proporciona Launchpad para albergar repositorios personales, permitiendo que los usuarios puedan agregar este repositorio a su archivo `/etc/apt/sources.list` o a sus orígenes de software si utilizan la aplicación gráfica correspondiente, y beneficiarse de disponer de las últimas versiones de la aplicación en cuanto estén disponibles, igual que se hace con otros paquetes disponibles en la distribución.

4.7.2. Diseño

Control de TempusFugit y Squealer desde el *applet*

En primer lugar, para llevar a cabo este cambio, será preciso modificar el menú contextual del *applet*, agregando las opciones necesarias para iniciar y detener tanto TempusFugit como Squealer.

En segundo lugar, será preciso agregar clases al paquete del *applet* que permitan gestionar la ejecución de los comandos `tempusfugit` y `squealer` desde código Python. Para ello, se crearán dos niveles de clases de control. El primero de ellos —y de más bajo nivel de abstracción— se encargará de lanzar un comando desde Python, manejando posibles errores que se produzcan durante la ejecución. El segundo nivel, que sólo se empleará para gestionar la ejecución de TempusFugit, trabajará internamente con el primer nivel, y ofrecerá una interfaz más rica, permitiendo conocer el estado de la ejecución o detenerla a voluntad.

Aunque esta diferenciación puede no ser trivial a simple vista, debe realizarse, ya que Squealer y TempusFugit son programas bien distintos. El primero de ellos se ejecutará puntualmente y de forma muy breve, por lo que no es tan importante brindar la posibilidad al usuario de cancelar la ejecución del programa. En el segundo caso, el

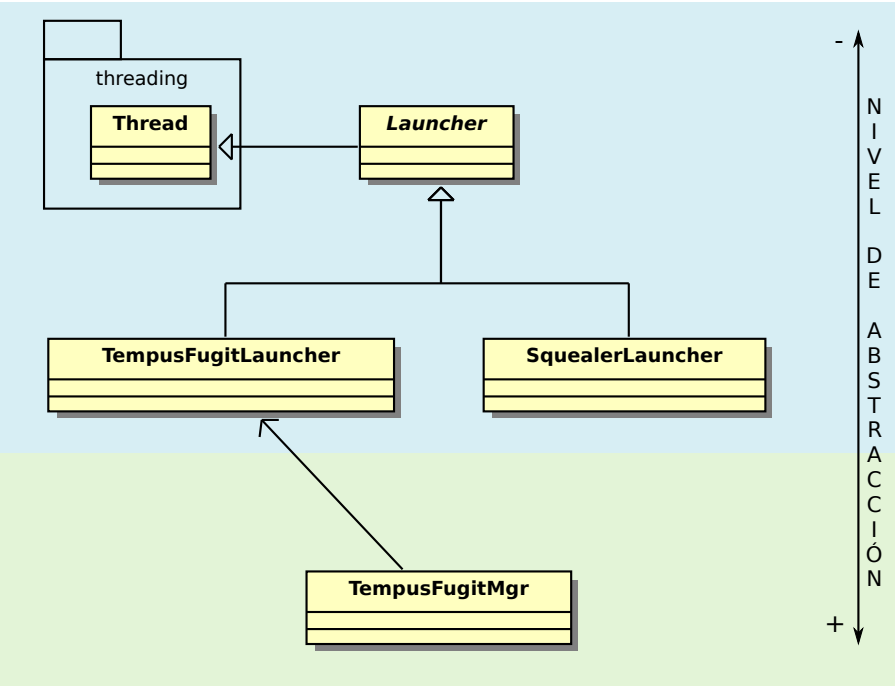

Figura 4.6: Niveles de clases para ejecutar procesos.

programa se ejecuta como un *demonio*[4], siendo imprescindible que el usuario tenga la posibilidad de iniciar, detener y conocer el estado de su ejecución en todo momento.

En lo que respecta a la interfaz de usuario, un factor clave a la hora de diseñarla es dedicar mucha atención a la interacción con el usuario. La interfaz debe ofrecer al usuario una manera simple e intuitiva para iniciar y detener las aplicaciones, al mismo tiempo que ofrece un nivel de información adecuado, sin llegar a saturar con muchos datos o molestar al usuario.

Para cumplir con las premisas anteriores, se ha pensado en, además de agregar dos opciones en el menú contextual que permitan manejar TempusFugit y Squealer, realizar los siguientes cambios:

- Permitir iniciar y detener TempusFugit haciendo un clic sobre el *applet*. Se ha determinado que esta acción será la más empleada por los usuarios, por lo que este comportamiento facilitará realizar esa acción.

- Indicar en todo momento el estado de la ejecución de TempusFugit, haciendo que el *applet* muestre en el panel de GNOME un icono si está ejecutando y otro

[4]Un demonio es un tipo especial de proceso informático que se ejecuta en segundo plano en vez de ser controlado directamente por el usuario (es un proceso no interactivo).

distinto si está detenido. Se ha escogido este modo de monitorizar TempusFugit para, por un lado, permitir que el usuario tenga realimentación cuando ejecute este programa y, por otro, que conozca en todo momento el estado de su ejecución con una simple mirada.

- Se ha decidido, que en caso que la ejecución de Squealer y/o TempusFugit concluya de forma exitosa, no se indicará nada al usuario, de modo que no se le molestará con mensajes posiblemente innecesarios[5]. De producirse cualquier error se alertará al usuario con un cuadro de diálogo indicando el problema detectado.

Visualización de resultados locales

El primer paso es estudiar el método empleado por DeTraS para almacenar los ficheros que incluyen los datos recogidos, que se detalla a continuación:

1. Cada vez que está en ejecución, TempusFugit almacena los eventos que va recogiendo en un único fichero XML.

2. Cuando el usuario envía los datos al servidor, Squealer hace una copia comprimida del fichero donde se almacenan los eventos, dejando el original vacío.

Debido a este peculiar método de almacenamiento de datos, se ha decidido mostrar únicamente los eventos existentes en el documento XML que aún no ha sido subido al servidor, correspondientes a los últimos eventos que han sido recogidos en el sistema del usuario y no han sido enviados aún al servidor.

Se ha decidido realizar una interfaz gráfica muy simple para mostrar los resultados que contiene el fichero de eventos, permitiendo al usuario ordenar los mismos por cada uno de sus campos. De este modo, se podrán conocer datos como los últimos eventos registrados, las aplicaciones más usadas, etcétera.

Con el fin de mostrar la información necesaria sobre los eventos en la ventana de la aplicación, se va a desarrollar un *parser* XML que permita, a partir del fichero que almacena todos los eventos registrados por TempusFugit, obtener la información que pueda resultar útil de los mismos.

Además, se agregará otra entrada al menú contextual del *applet* de GNOME para poder lanzar desde allí la ventana previamente mencionada.

[5]Como podrá leer en el apartado 4.8, este comportamiento fue alterado a posteriori gracias a la información recibida por usuarios de la aplicación

Mejoras en la configuración de DeTraS

Con el fin de resolver los problemas detectados en la iteración anterior sobre el sistema desarrollado para emplear la configuración —y que se comentaron brevemente en la sección 4.7.1—, se van a realizar las siguientes modificaciones:

1. Eliminar la clase `GlobalConfig` del paquete de *Squealer*.
2. Crear dos nuevas clases, `SettingsReader` y `SettingsWritter`, que se encarguen abstraer la lectura y escritura de las preferencias.
3. A partir de ahora, ninguna clase leerá o escribirá la configuración directamente empleando la clase `SettingsManager` salvo las dos mencionadas en el punto anterior.

Tomando estas medidas se permite que cualquier objeto pueda leer y guardar las preferencias sin peligros de concurrencia y se deja en manos de las clases `SettingsReader` y `SettingsWritter` la tarea de manejar las preferencias permitiendo, además, que si en algún momento se efectúa algún cambio en la forma de almacenarlas, el resto de clases no se vean afectadas.

Ampliación de las opciones de configuración

De cara a conseguir ampliar las opciones disponibles para el usuario como se comentó en la sección 4.7.1, se deben agregar dos entradas más en el fichero de configuración, así como modificar la interfaz de usuario que gestiona las preferencias, permitiendo al usuario alterar los valores de esas dos preferencias.

4.7.3. Implementación

Control de TempusFugit y Squealer desde el *applet*

En primer lugar, se han agregado las entradas pertinentes al menú contextual del *applet* para poder lanzar la ejecución de TempusFugit y Squealer desde el mismo.

Posteriormente, se han desarrollado los dos niveles de clases para permitir la gestión de TempusFugit y Squealer desde Python —expuestos en la figura 4.6—. El primer paso fue desarrollar las clases `Launcher`, `SquealerLauncher` y `TempusFugitLauncher`, que se encargan de lanzar la ejecución de Squealer y TempusFugit en un

```
1  class Launcher (threading.Thread):
2      """Generic class to control commands execution"""
3      def __init__ (self, parent):
4          threading.Thread.__init__(self)
5          self.parent = parent
6
7      def launch (self):
8          pass
9
10     def run (self):
11         self.launch ()
```

Listado 4.6: Clase `Launcher`.

nuevo thread y capturar cualquier problema que pueda surgir en su ejecución para notificarla.

En el listado 4.6 se puede observar el procedimiento para crear una clase Python que utilice un nuevo hilo de ejecución para llevar a cabo sus tareas. El mecanismo para crear *threads* en Python es muy sencillo: por un lado, la clase que debe ejecutar en un nuevo hilo debe heredar de la clase `threading.Thread`; por otro lado, debe implementar el método `run`, que será el ejecutado en el nuevo hilo; finalmente, para llevar a cabo la ejecución en el nuevo hilo, se debe llamar al método *start* de una instancia de la clase —heredado de `threading.Thread`—.

Si estudia el listado 4.7, observará la forma en que se lanza un proceso desde código Python. La clase `subprocess.Popen` es la encargada de ello, además de permitir trabajar con la entrada y la salida del proceso a ejecutar. El código para lanzar la ejecución de TempusFugit es muy similar al mostrado en el listado indicado.

El siguiente nivel de abstracción —donde se encuentra `TempusFugitMgr`—, emplea las clases del nivel inferior, junto al fichero que contiene el PID[6] de la ejecución de TempusFugit —que se puede encontrar en la ruta `/tmp/.detras/tempusfugit-<nombre-usuario>.pid` para ofrecer la funcionalidad que se muestra en su diagrama de clase, en la figura 4.7.

Una vez completados los cambios anteriores, es hora de centrarse en conectar estas

[6]El identificador de procesos —o PID por sus siglas en inglés— es un número entero usado por el *kernel* de algunos sistemas operativos (como el de Unix o el de Windows NT) para identificar un proceso de forma unívoca.

```
1  class SquealerLauncher (Launcher):
2      """Handle␣Squealer␣execution"""
3
4      def __init__ (self, parent):
5          Launcher.__init__ (self, parent)
6          self.popen = None
7
8      def launch (self):
9          try:
10             path = os.path.join (config.BIN_DIR, 'squealer')
11             self.popen = subprocess.Popen ([path], stderr=subprocess.
                   PIPE)
12
13             stderr = self.popen.communicate (None) [1]
14
15             if (stderr is not None) and (stderr != ""):
16                 self.parent.error_cb ("Squealer␣error", stderr)
17         except (OSError, ValueError), e:
18             self.parent.error_cb ("Error", \
19                     "Cannot␣execute␣Squealer:␣\n␣%s" % str(e))
20         except:
21             self.parent.error_cb ("Error", \
22                     "Cannot␣execute␣Squealer:␣\n␣Unexpected␣error")
23         finally:
24             self.parent.upload_finished_cb ()
```

Listado 4.7: Clase `SquealerLauncher`.

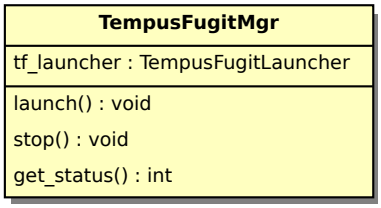

Figura 4.7: Diagrama de la clase `TempusFugitMgr`.

```
1  class DetrasApplet (object):
2      def __init__ (self, applet):
3          (...)
4          self.applet.connect ("button-press-event", self.
               on_button_press_cb)
5          (...)
6
7      def on_button_press_cb (self, widget, event):
8          if event.button == 1:
9              self.toggle_tf_cb (None, None)
10
11     def toggle_tf_cb (self, component, verb):
12         if self.tf.get_status () is TempusFugitMgr.RUNNING:
13             self.tf.stop ()
14         else:
15             self.tf.launch ()
```

Listado 4.8: Código que permite ejecutar TempusFugit al pulsar sobre el *applet*.

modificaciones con la interfaz de usuario. Si nos centramos en la parte del código que hace que TempusFugit se inicie o detenga al pulsar sobre el icono del *applet*, veremos que, para comenzar, es preciso escuchar el evento que se lanzará al pulsar sobre el *applet*, al que debemos asociar una función a la que se llamará cuando salte ese evento. El código que realiza este proceso se encuentra en el listado 4.8.

La actualización del icono del *applet* guarda ciertas similitudes con la forma de ejecutar TempusFugit al pulsar sobre el icono del *applet*. En este caso, generamos —gracias al método `timeout_add` de GObject— un evento que se lanzará cada determinado tiempo, que es preciso especificar. Entonces, basta con asociar a ese evento un método que se encargue de comprobar el estado de TempusFugit y actualizar el icono, si ese estado ha cambiado, para completar la funcionalidad.

Finalmente, con el fin de mostrar los posibles errores encontrados en la ejecución de TempusFugit y/o Squealer, los métodos que ejecutan los procesos, en caso de encontrar algún problema, llaman a métodos que se han agregado a la clase `DetrasApplet` y que emplean la clase `gtk.MessageDialog` para mostrar un error. El código de este diálogo de error es similar al que se muestra en el listado 4.10.

Visualización de resultados locales

En primer lugar, se ha agregado una entrada al menú contextual del *applet* para poder lanzar la ventana de visualización de resultados locales desde él.

A continuación, se ha creado la clase `Event`, que representa un evento almacenado por TempusFugit. En el listado de eventos mostrado al usuario sólo se incluirán los eventos de tipo *WindowChange*, es decir, los eventos ocurridos al cambiar de ventana, que son los que van a interesar al usuario. Los otros tipos de eventos se emplearán para realizar cálculos de duración.

Asimismo, se ha creado la clase `EventsParser`, que permite *parsear* el fichero con los eventos registrados y crea un objeto de la clase `Event` para aquellos que sean de tipo *WindowChange*. Puede observar cómo se realiza el proceso de *parseado* en el listado 4.9.

```
1     def parse (self):
2         """Convert␣an␣XML␣file␣containing␣events␣in␣an␣Events'␣list""
              "
3         self.locker.lock (self.events_file)
4         try:
5             xml_doc = minidom.parse (self.events_file)
6             self.locker.unlock (self.events_file)
7
8             events_nodes = xml_doc.getElementsByTagName ("event")
9             events = []
10
11            for i in events_nodes:
12                if (i.hasAttribute ("type")) \
13                        and (i.getAttribute ("type") == "WindowChange
                              "):
14                    i.normalize ()
15
16                    itime = i.getAttribute ("time")
17
18                    next_event = i.nextSibling
19
20                    while (next_event is not None) \
21                            and (next_event.nodeType != Node.
                                  ELEMENT_NODE):
22                        next_event = next_event.nextSibling
23
24                    if next_event is None:
25                        etime = int (time.time ())
```

```
26                      else:
27                          etime = next_event.getAttribute ("time")
28
29                      app_child = i.getElementsByTagName ("app") \
30                              [0].firstChild
31
32                      if app_child is not None:
33                          app = app_child.data
34
35                      title_child = i.getElementsByTagName ("title") \
36                              [0].firstChild
37
38                      if title_child is not None:
39                          title = title_child.data
40
41                      events.append (Event (itime, etime, app, title))
42
43              xml_doc.unlink ()
44
45              return events
46          except IOError:
47              return ""
```

Listado 4.9: Código para *parsear* el XML de eventos.

Finalmente, se ha procedido a crear la clase `OverviewDialog`, que permite mostrar el listado de eventos, y se ha conectado esta clase con la acción correspondiente del menú contextual del *applet*.

Mejoras en la configuración de DeTraS

Se ha comenzado esta tarea creando las clases `SettingsReader` y `SettingsWritter`, que van a permitir leer y escribir las preferencias. Como esta cuestión se asemeja bastante al clásico problema del lector y el escritor[7], en este apartado llamaré a `SettingsReader` lector y a `SettingsWritter` escritor. Las diferencias entre ambas clases son las siguientes:

1. El lector únicamente permite leer las preferencias, mientras que el escritor permite realizar tanto lectura como escritura.

2. Sólo podrá existir en el mismo momento una instancia de la clase escritor.

[7]Puede leerse más al respecto en cualquier libro de sistemas operativos o aquí: `http://www.infor.uva.es/~cllamas/concurr/pract98/sisos30/index.html`

3. El escritor necesitará bloquear la lectura y/o modificación del fichero hasta que termine su tarea, impidiendo hasta entonces la creación de nuevos lectores. Cabe resaltar que los lectores creados con anterioridad podrán seguir funcionando, ya que cuando se instancian leen el fichero por completo por motivos de eficiencia y para evitar posibles incongruencias en la configuración. Un ejemplo de incongruencia en las preferencias se daría cuando el usuario ha marcado la opción para filtrar los datos con expresiones regulares pero aún no ha introducido ninguna. Otro ejemplo ocurriría si el usuario está escribiendo su nombre de usuario pero aún no ha concluido.

Una vez implementado el comportamiento descrito, se modificaron las clases que usaban la configuración anterior para adaptarlas a las mejoras indicadas.

Ampliación de las opciones de configuración

En primer lugar, se han agregado las dos opciones mencionadas al fichero de configuración mostrado en el listado 4.2.

Una vez realizado ese cambio, se ha modificado la interfaz gráfica de las preferencias usando Glade y se ha implementado en la clase `PrefsDialog` el código necesario para manejar estas preferencias.

4.8. Quinta iteración

4.8.1. Especificación

Visualización de datos globales

El primer objetivo de esta iteración es el desarrollo de una herramienta para visualizar los datos globales recopilados por DeTraS a través de un interfaz web. Esta aplicación será llamada Dazer, que proviene del acrónimo del inglés ***Da**ta Analyz**er*** —o Analizador de Datos en castellano—.

Esta herramienta tiene, en principio, carácter académico, aunque puede apreciarse su potencial tanto en un ambiente corporativo, como para ofrecer datos interesantes para cualquier usuario de Internet. La funcionalidad básica que debe presentar la aplicación es la siguiente:

1. Proporcionar información útil sobre el uso de aplicaciones, el trabajo de los clientes y el trabajo realizado en los distintos proyectos a partir de los datos recopilados y almacenados por DeTraS.

2. Mostrar la evolución de uso de una aplicación a lo largo del tiempo.

3. Mostrar la evolución del trabajo realizado en un proyecto a lo largo del tiempo.

4. Ofrecer una interfaz de usuario intuitiva y fácil de usar permitiendo, a su vez, el filtrado y ajuste de los datos que muestra la aplicación, de modo que el usuario de la misma sea capaz de encontrar lo que necesita.

En mi opinión, todo sitio web desarrollado en la actualidad debe estar diseñado para ceñirse a los estándares web, ofreciendo servicio al menos a una gran mayoría de los navegadores web disponibles. Por tanto, además de las funcionalidades mencionadas con anterioridad, la aplicación debe respetar los estándares web que rigen el buen funcionamiento de las mismas.

Mejora de realimentación en el *applet*

En segundo lugar, y aunque este cambio no estaba previsto, después de recibir algunas sugerencias sobre la versión resultado de la cuarta iteración y estudiarlas con detenimiento, he considerado oportuno realizar una modificación en el código del *applet* para ofrecer al usuario una mejor realimentación.

Revisión y actualización de la documentación

Asimismo, también se considera objetivo de esta iteración actualizar y retocar el boceto de manual de usuario realizado en la iteración anterior, ofreciendo su descarga independientemente de los paquetes de instalación de la aplicación e incluyéndolo junto al código fuente de DeTraS. Así, los usuarios podrán disponer de un documento fácil de entender, y en un formato accesible, que les enseñe rápidamente a manejar la aplicación correctamente.

Facilitar la instalación de la aplicación

Finalmente, quería destacar que en esta iteración se ofrecerá nuevamente la opción de instalar la aplicación desde paquetes Debian y el repositorio del proyecto, de modo

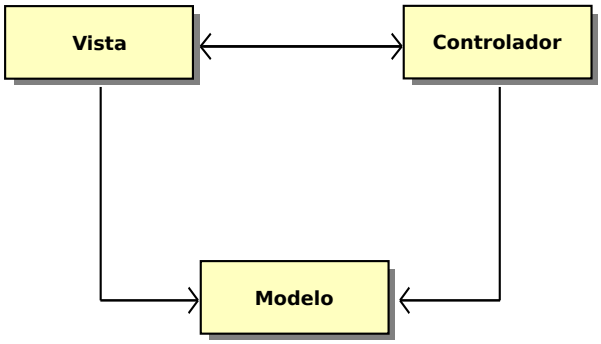

Figura 4.8: Diagrama MVC.

que se consiga el objetivo de facilitar la instalación de la aplicación.

4.8.2. Diseño

Visualización de datos globales

Dazer es desarrollado empleando el *framework* de desarrollo web Django, del que hablé en la sección 2.5.6, y que obliga a seguir un patrón MVC[8] para el desarrollo de la aplicación web.

De este modo, el objetivo fundamental de esta fase será discernir cómo dividir la aplicación correctamente en los tres componentes del patrón.

Modelo. Este punto es el más claro, ya que la información persistente va a ser la ya existente en la base de datos de DeTraS, y no es necesario agregar o modificar nada en ella. Como esta aplicación trabajará con los datos existentes, se crearán las clases que sean necesarias para manejarlos, cuyo diagrama puede observarse en la figura 4.9.

Vista. Este componente permitirá al usuario interactuar con el sistema, por lo que es el lugar donde se debe trabajar para ofrecer una interfaz intuitiva, amigable y que responda adecuadamente a los cambios producidos en el sistema. En este caso, al tratarse de una aplicación web, la interfaz gráfica consistirá en una serie de documentos HTML —complementados con otras tecnologías— que permiten al usuario manejar el sistema. Al igual que el resto de interfaces gráficas desarrolla-

[8]Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

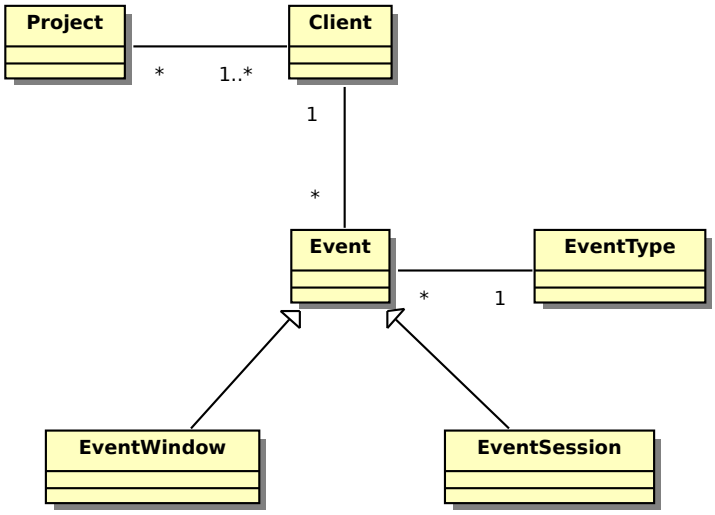

Figura 4.9: Diagrama de clases del modelo de Dazer.

das a lo largo del presente proyecto, se ha ido evolucionado el diseño de la web empleando bocetos.

Controlador. En esta parte del modelo residirá la lógica de la aplicación, que es donde habrá que centrar los esfuerzos en conseguir obtener datos valiosos para el usuario a partir de la información que proporcione el mismo. Las clases que se crearán en el controlador pueden examinarse en la figura 4.10.

Mejora de realimentación en el *applet*

La mejora consistirá en mostrar al usuario una ventana de confirmación cuando sus datos hayan sido cargados correctamente en el servidor de DeTraS. Este cambio deberá realizarse en la clase `DetrasApplet`, que ya he mencionado anteriormente.

4.8.3. Implementación

Visualización de datos globales

En primera instancia, se han definido las clases del modelo indicadas en el apartado 4.8.2. Estas clases representan los objetos almacenados en la base de datos, por lo que se emplearán —junto a algunas sentencias SQL para mejorar el rendimiento— para manejar los datos de la misma.

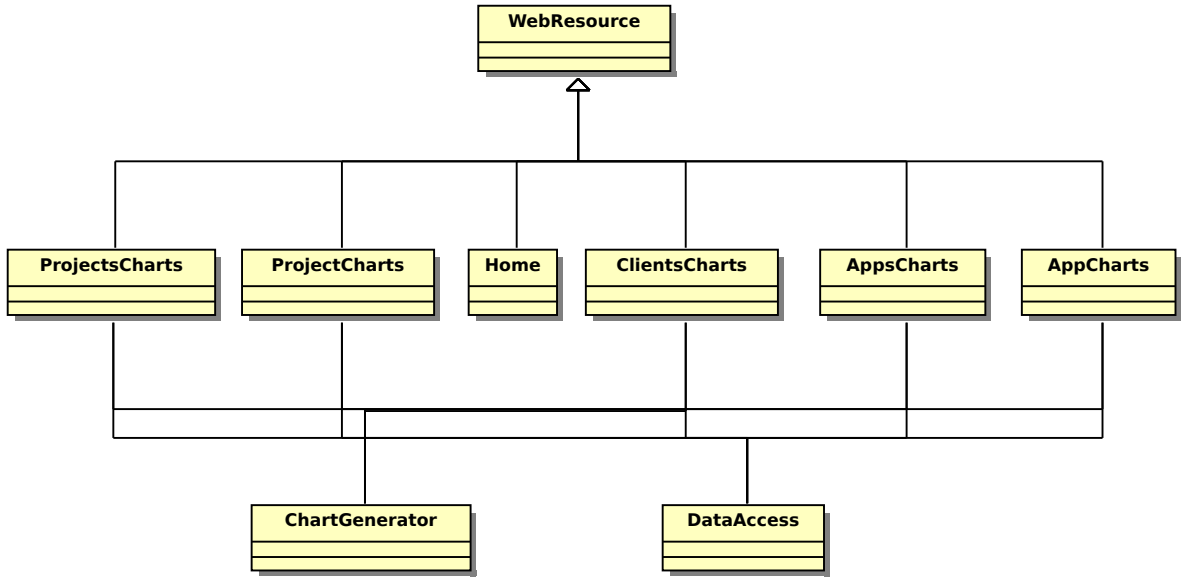

Figura 4.10: Diagrama de clases del controlador de Dazer.

A continuación, se definieron las URL[9] de acceso a las distintas páginas que tendrá la aplicación, implementando a su vez las clases que entrarán en juego cuando se acceda a esas URL y que forman parte del controlador. Cabe destacar que la clase `WebResource`, de la que heredan el resto de clases que conforman la lógica de la aplicación salvo `ChartsGenerator` y `DataAccess`, permite a las clases que la extienden soportar los distintos métodos definidos por HTTP, es decir, los métodos *GET*, *POST*, *PUT* y *DELETE*, así como manejar parámetros que se incluyan dentro de las propias URL de la aplicación. Al mismo tiempo se desarrolló el código necesario para generar las gráficas a partir de los datos existentes en la base de datos —código que se encuentra en la clase `ChartsGenerator`—, además del código para acceder al modelo de la aplicación, que se halla en la clase `DataAccess`.

Finalmente, se plasmaron en código HTML las distintas páginas web que conforman la aplicación, ayudándose de otras tecnologías para ofrecer una apariencia más vistosa y agradable.

[9]Un localizador uniforme de recursos, más comúnmente denominado URL —*Uniform Resource Locator*—, es una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se usa para nombrar recursos en Internet para su localización o identificación, como por ejemplo documentos textuales, imágenes, vídeos, presentaciones digitales, etc.

```
1     def __show_confirm (self, title, message):
2         gtk.gdk.threads_enter ()
3
4         msg = gtk.MessageDialog (None, \
5                 gtk.DIALOG_MODAL | gtk.DIALOG_DESTROY_WITH_PARENT, \
6                 gtk.MESSAGE_INFO, gtk.BUTTONS_OK, message)
7
8         msg.set_title (title)
9
10        msg.run ()
11        msg.destroy ()
12
13        gtk.gdk.threads_leave ()
```

Listado 4.10: Código para mostrar la ventana de confirmación.

Mejoras en el *applet*

Para conseguir mostrar la ventana de confirmación cuando la información del usuario se haya subido correctamente, es necesario agregar el método `__show_confirm` a la clase `DetrasApplet`, que se encargará de emplear un objeto de la clase `gtk.MessageDialog` para mostrar un diálogo de información, como puede verse en el listado 4.10.

Finalmente, también es preciso modificar el método `upload_finished_cb` de la propia clase `DetrasApplet` para que llame al anterior.

Capítulo 5

Resultados Experimentales

Una vez se ha terminado de escribir la aplicación, es necesario realizar comprobaciones para asegurarse que cumple con su función.

Desde que se lanzó la versión inicial de DeTraS —que corresponde con el resultado de la primera iteración detallada en el apartado 4.4—, ha existido a disposición de los usuarios un servidor ejecutando el servicio web de recogida de datos de DeTraS. Todos estos datos recopilados constituyen una buena fuente de información para demostrar que la aplicación funciona, por lo que serán empleados para tal tarea. Quisiera recalcar que el fin del presente proyecto no consiste en el análisis de los datos registrados por la aplicación, sino que son empleados únicamente como prueba del funcionamiento de DeTraS. Como puede imaginarse el lector, existen múltiples formas de abordar esos datos. Con el fin de comprobar todo el funcionamiento del sistema, se muestran a continuación varios enfoques ofrecidos por Dazer, mostrando la información y las gráficas de los mismos.

5.1. Aplicaciones

En este apartado se muestran varios ejemplos con la información referente al uso de aplicaciones que se extrae de los datos recogidos.

5.1.1. Aplicaciones más utilizadas

En este ejemplo pueden observarse las diez aplicaciones utilizadas durante más tiempo en los treinta últimos días. Como puede apreciarse en los datos recogidos en la tabla 5.1 y la gráfica de la figura 5.1, la aplicación más usada durante ese periodo de

Aplicación	Tiempo de uso (en horas)
Google-chrome	73.465
Evolution	50.263
Chromium-browser	28.200
Terminator	24.219
Evince	18.143
Winefish	15.960
Emacs	8.520
Gnome-terminal	6.180
OpenOffice.org 3.0	3.923
Gedit	2.203

Tabla 5.1: Aplicaciones más empleadas entre el 17/05/2010 y el 16/06/2010.

Mes	Tiempo de uso (en horas)
12	0.000
1	0.000
2	0.000
3	0.720
4	10.600
5	30.518
6	16.409

Tabla 5.2: Evolución de uso de Chromium–browser.

tiempo fue el navegador Google Chrome, que cuenta con setenta y tres horas de uso, seguido del cliente de correo Evolution, que fue empleado durante cincuenta horas.

5.1.2. Evolución de una aplicación

Este ejemplo muestra la evolución de uso de un programa, concretamente el navegador Chromium, tanto con los datos recogidos —disponibles en la tabla 5.2— como su representación gráfica —en la figura 5.2—.

5.2. Clientes

Otra información que la aplicación puede mostrar, y puede resultar útil, es el trabajo realizado por los distintos usuarios de la aplicación. Por cuestiones de privacidad, en lugar de mostrar el nombre de usuario o su identificador, se genera un número para

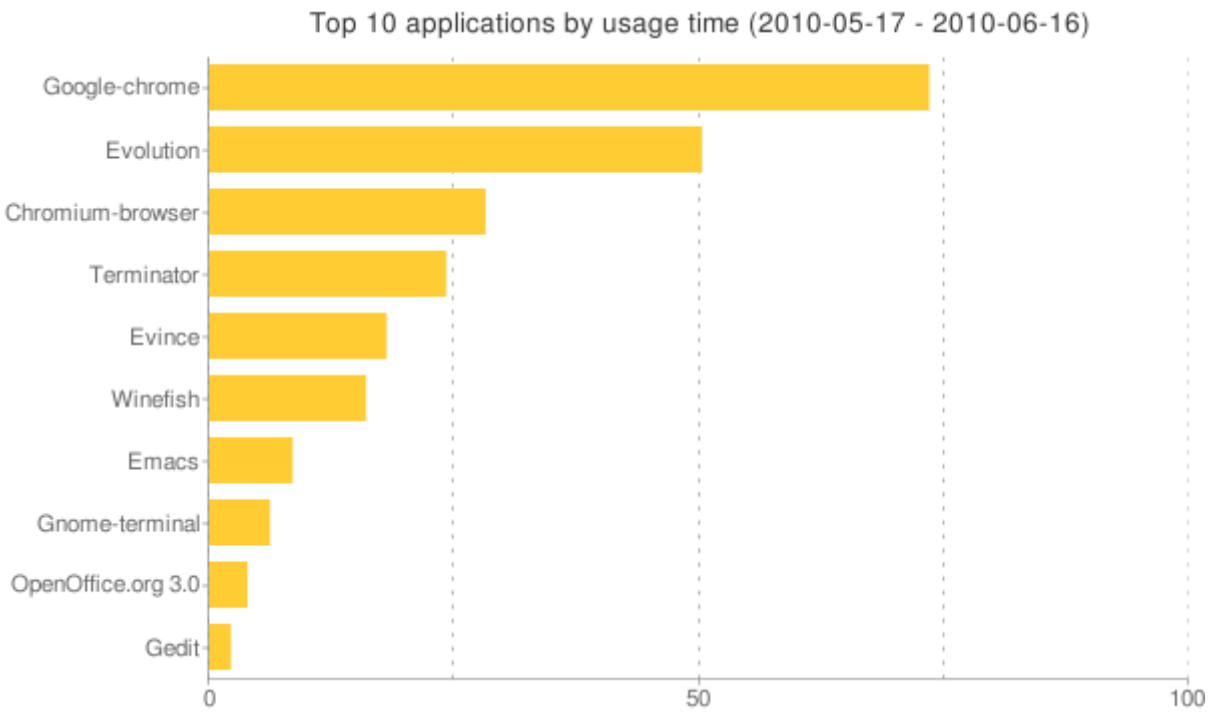

Figura 5.1: Aplicaciones más empleadas entre el 17/05/2010 y el 16/06/2010.

Cliente	Tiempo trabajado (en horas)
Client 4	209.801
Client 1	189.926
Client 5	3.429
Client 2	0.002
Client 3	0.001

Tabla 5.3: Clientes más activos entre el 01/03/2010 y el 15/06/2010.

cada uno de los usuarios que se van a mostrar en los análisis.

En esta ocasión, vamos a analizar los usuarios más activos desde el uno de marzo de 2010 hasta el quince de junio de 2010. Como puede advertir en la tabla 5.3 y la figura 5.3, únicamente cinco usuarios han reportado sus datos al servidor de DeTraS. Entre ellos, destacan especialmente dos usuarios, habiendo empleado la aplicación unas doscientas diez y ciento noventa horas respectivamente.

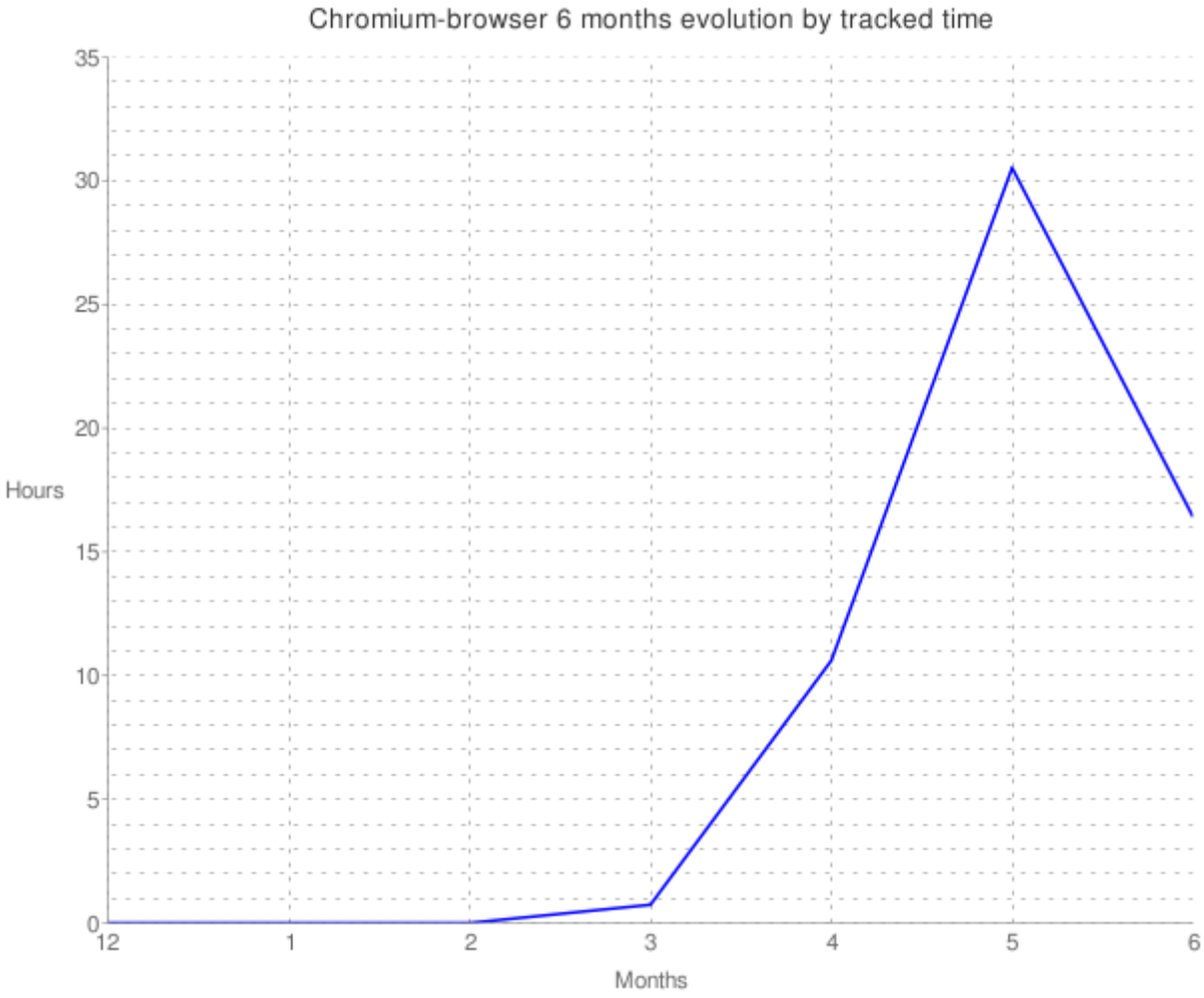

Figura 5.2: Evolución de uso de Chromium–browser.

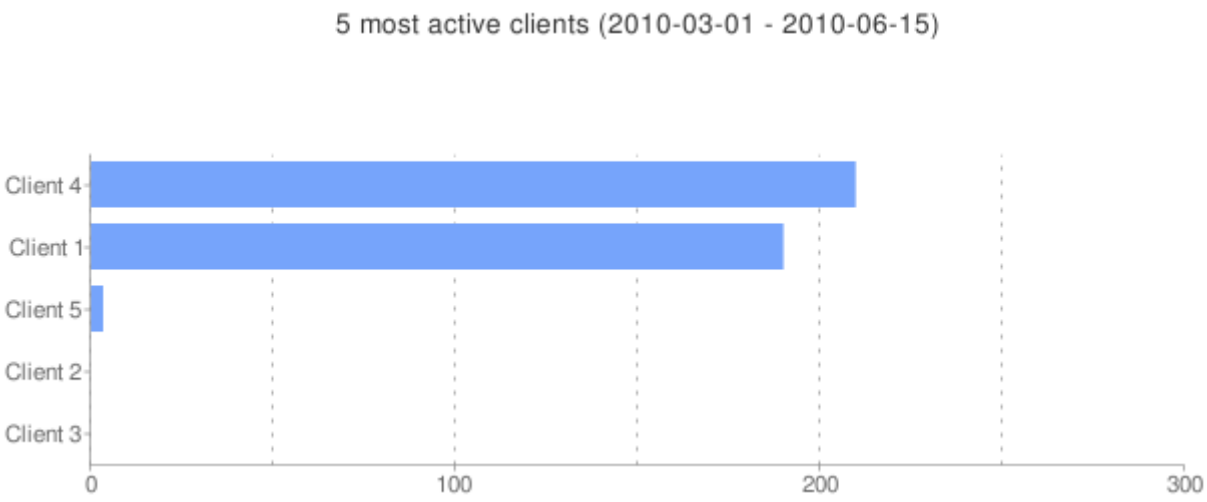

Figura 5.3: Clientes más activos entre el 01/03/2010 y el 15/06/2010.

Proyecto	Número de miembros
DeTraS	1

Tabla 5.4: Proyectos con mayor número de miembros.

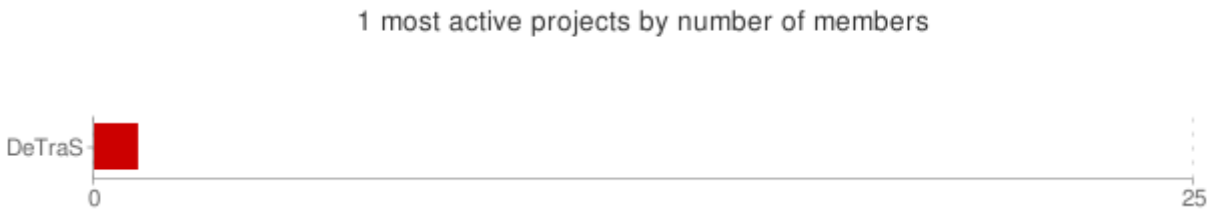

Figura 5.4: Proyectos con mayor número de miembros.

5.3. Proyectos

El tercer modo de enfocar los datos que provee Dazer consiste en ofrecer un análisis de la información de los distintos proyectos en los que trabajan los usuarios. Como puede presuponer, un usuario no tiene porqué pertenecer a ningún proyecto.

5.3.1. Proyectos con mayor número de miembros

Este ejemplo permite observar los proyectos que tienen un mayor número de miembros. Desgraciadamente, el único usuario que ha indicado hasta ahora el proyecto en el que está trabajando soy yo, así que DeTraS es el único proyecto que aparece, como puede comprobar en la tabla 5.4 y la figura 5.4.

5.3.2. Evolución de un proyecto

Es posible visualizar la evolución de un proyecto a lo largo del tiempo, como ocurre en este apartado con DeTraS. Los datos relativos al trabajo realizado en los seis últimos meses de DeTraS pueden encontrarse en la tabla 5.5, así como su representación gráfica en la figura 5.5.

Al tratar este apartado sobre el proyecto que se ha desarrollado, voy a detenerme un momento para comentar más a fondo la información mostrada.

En primer lugar, he de indicar que los datos indicados en este apartado no son totalmente correctos, ya que en una buena fase del desarrollo ha sido imposible utilizar el

Mes	Tiempo trabajado (en horas)
12	0.000
1	0.028
2	7.215
3	10.564
4	64.876
5	63.319
6	51.167

Tabla 5.5: Evolución de trabajo en DeTraS.

programa a la vez que se desarrollaba, y en los comienzos del proyecto no se registraba ninguna información sobre el mismo.

Como puede observar en la gráfica de la figura 5.5, el registro de datos del desarrollo de DeTraS comenzó el mes de enero, aunque apenas se recogieron datos entonces. Posteriormente, el número de horas dedicadas al proyecto fue ascendiendo hasta alcanzar su máximo en abril. Desde entonces, se puede apreciar que el número de tiempo dedicado al mes se mantiene en torno a sesenta y cinco horas mensuales, sin contar este mes de junio, que aún no ha terminado.

5.4. Conclusiones

Como ha podido observar en los distintos apartados pertenecientes a este capítulo, el sistema desarrollado en este proyecto está recogiendo datos y, a partir de los mismos, es capaz de ofrecer una serie de información de utilidad referente a las aplicaciones empleadas, el trabajo realizado por los usuarios y el trabajo realizado en los distintos proyectos.

En lo que respecta al volumen de datos con el que se han realizado las conclusiones, quería resaltar que, pese a ser bastante bajo —en parte debido al poco tiempo que lleva el proyecto en funcionamiento y a la base de usuarios prácticamente inexistente con la que se contaba en un inicio—, se han recogido suficientes datos como para verificar el funcionamiento de la aplicación.

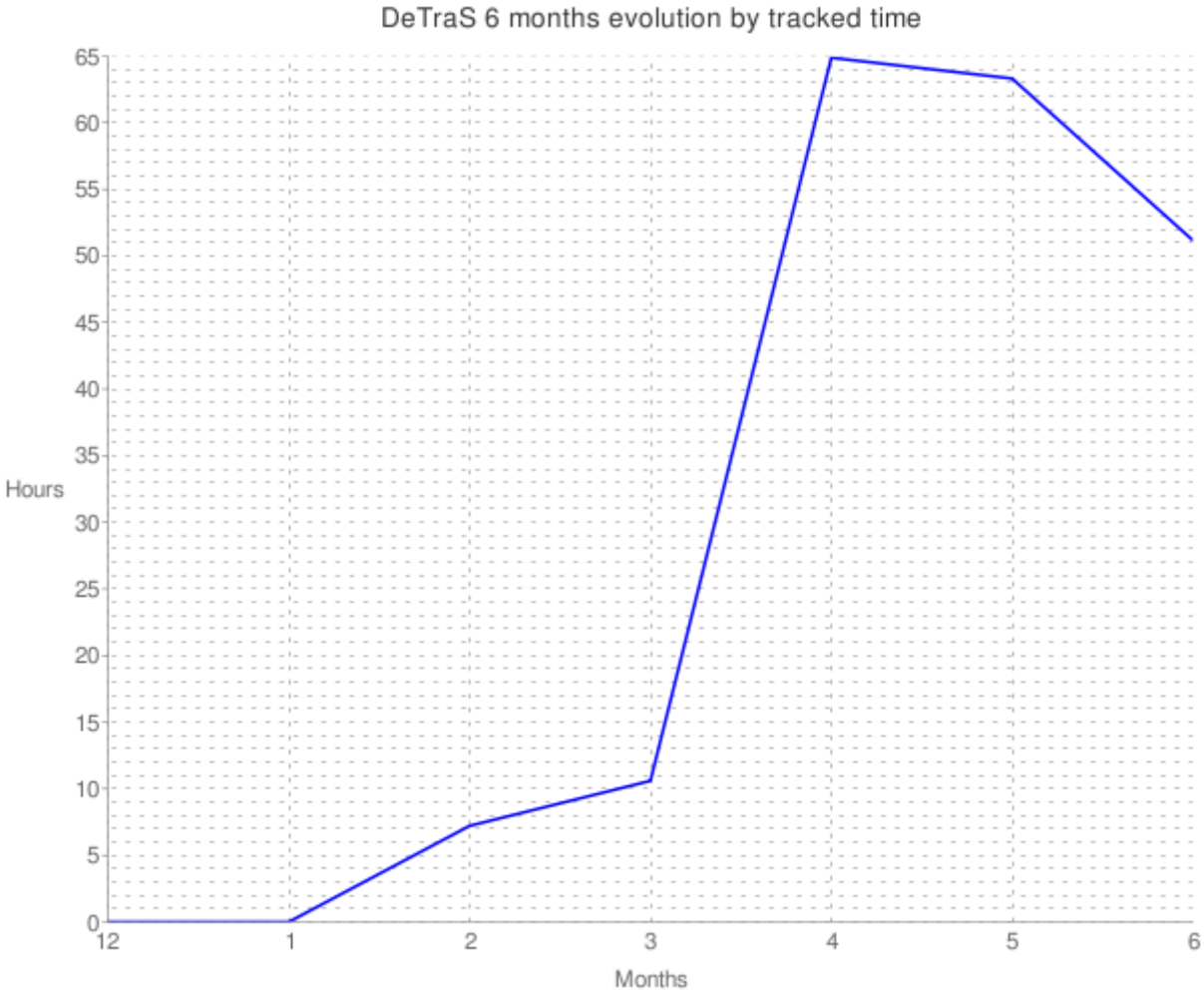

Figura 5.5: Evolución de trabajo en DeTraS.

Capítulo 6

Conclusiones

Una vez realizado el desarrollo y analizados los resultados del mismo, es momento de realizar un breve repaso por los logros alcanzados, así como extraer conclusiones de todo el proceso descrito en este documento.

6.1. Logros alcanzados

El presente proyecto pretendía ampliar el sistema DeTraS, dotándolo de nuevas funcionalidades y mejorando las existentes, logrando desarrollar un producto software funcional que permitiese realizar un seguimiento de las tareas realizadas por un usuario en su ordenador.

Como puede comprobarse, se han completado todos los objetivos marcados al inicio el proyecto —detallados en el capítulo 3— de forma exitosa. A continuación se ofrece un breve resumen de los hitos alcanzados:

Detección de inactividad mediante el uso del salvapantallas. Se ha mejorado la detección de inactividad de DeTraS, añadiendo la detección del estado del salvapantallas —en concreto, de GNOME Screensaver—. Esta funcionalidad permite que la información recogida en el ordenador del usuario es más precisa.

Filtrado de la información recogida. Con el fin de mantener la privacidad de los usuarios de DeTraS, se ha desarrollado un sistema de filtrado de la información recogida en el sistema del usuario. Este sistema permite escoger entre varios niveles de privacidad, lo que ha provocado un cambio radical en el método en el que se estaba llevando a cabo la configuración de DeTraS.

Desarrollo de un nuevo *applet*. Dado que el *applet* antiguo de DeTraS ofrecía una pobre funcionalidad y se habían detectado algunos fallos en él, se ha desarrollado un nuevo *applet* para GNOME que permite:

Visualizar los datos locales. Se ofrece una interfaz gráfica de usuario muy sencilla en la que pueden observarse los últimos eventos registrados por DeTraS.

Configuración. Otra de las mejoras llevadas a cabo consiste en permitir configurar los parámetros de la aplicación desde el propio *applet*, desde una interfaz gráfica desarrollada para ese menester.

Activar y desactivar TempusFugit y Squealer. De este modo, el usuario puede comenzar o detener el seguimiento de su sistema desde el propio *applet*, así como enviar los datos recogidos al servidor de DeTraS. Asimismo, se muestra al usuario en todo momento el estado en el que se encuentra TempusFugit, de manera que sepa si se está monitorizando el uso de su sistema rápidamente.

Visualizar los datos globales. Para ofrecer información fácil de entender, útil y que pueda obtener cualquier usuario, se ha desarrollado la aplicación web Dazer. Esta herramienta ofrece al usuario la posibilidad de ver información sobre el uso de aplicaciones, el trabajo realizado por los clientes de DeTraS y los proyectos en los que trabajan.

Facilitar la instalación de la aplicación. Otro punto en el que se ha trabajado es en realizar una instalación del cliente de DeTraS más sencilla, permitiendo su instalación desde paquetes Debian y desde un repositorio. Asimismo se ha trabajado en la creación y mantenimiento de una variada y sencilla documentación de cara al usuario.

6.2. Conocimientos adquiridos

El desarrollo del presente proyecto me ha permitido adquirir conocimientos en varios campos, entre los que destaco:

Conocimientos en GLib. Aunque había trabajado previamente con C tanto en GNU/-Linux como en Plan 9, nunca lo había hecho con esta biblioteca de programación

ampliamente usada en el mundo GNU/Linux.

Ampliación de los conocimientos de Python, obteniendo una visión más amplia de sus bibliotecas, profundizando en su sintaxis y trabajando con herramientas relacionadas al lenguaje como Django.

Gestión de un proyecto de software libre. Desarrollar todo este proyecto como software libre me ha permitido comprender lo complicado que es gestionar un proyecto y la gran cantidad de decisiones que hay que tomar ante cualquier problema que surge.

Funcionamiento de D-Bus. Como usuario de GNU/Linux, hacía tiempo que había escuchado hablar de D-Bus, aunque hasta ahora no había tenido la posibilidad de emplearlo. Realizar el proyecto me ha permitido comprender mejor su funcionamiento y el modo en que se emplea.

Construyendo aplicaciones gráficas en GNU/Linux. Las escasas interfaces gráficas que he realizado a lo largo de la carrera han sido desarrolladas en Java o bien han sido interfaces web. Aunque el proceso de desarrollar interfaces gráficas es similar, en esta ocasión he podido emplear y conocer otro tipo de herramientas para desarrollarlas. Además, me ha sorprendido que la interfaz obtenida sea independiente del lenguaje de programación.

Procesos de instalación. Nuevamente, había creado y empleado alguna vez *Makefiles*, pero desarrollar este proyecto me ha dado la posibilidad de conocer más el proceso de instalación, además de crear mis primeros paquetes Debian y mantener mi primer repositorio de código.

Búsqueda de documentación. Algunas tareas requeridas para implementar el proyecto carecían prácticamente de documentación, lo que me ha obligado a buscar exhaustivamente y a leer bastante código de otros proyectos. Un buen ejemplo donde sucedió este problema fue en la creación del *applet* para GNOME, ya que, aunque hay varios manuales que explican los pasos más simples de la construcción del *applet*, no he encontrado una documentación clara sobre las opciones disponibles al intentar realizar algo más complejo.

6.3. Trabajos futuros

A continuación, se ofrece una lista de líneas de desarrollo que permitirían extender y mejorar las funcionalidades ofrecidas por el software implementado.

1. Ampliación de la herramienta Dazer, pudiendo llevarse a cabo de múltiples formas: permitiendo analizar los datos desde otros puntos de vista interesantes —por ejemplo, permitiendo consultar la información de cada usuario individualmente—; agregando herramientas sociales a la aplicación, de manera que puedan comentarse gráficas o interactuar con otros usuarios de distintas formas; o ampliando las opciones de filtrado a la hora de mostrar información.

2. Ampliación de las preferencias. Entre otras, podría ser especialmente interesante el ofrecer una opción que permita subir los datos usando Squealer cada cierto tiempo —indicado por el usuario—. Permitir modificar el tiempo tras el cual se considera que el usuario está inactivo también podría ser una buena opción.

3. Mejoras en el *applet*, como indicar el estado en el que se encuentra Squealer desde el icono presentado en el panel.

4. Ampliar la interfaz para mostrar datos locales. Brindar la posibilidad de visualizar datos enviados con anterioridad, incluir algún tipo de gráfica o estadística y permitir editar eventos son algunas de las características que podrían desarrollarse.

5. Mejorar la estructura del documento de eventos. Como se ha comentado anteriormente, la estructura del documento XML que almacena los eventos es ambigua. Crear una nueva estructura que valide con un esquema XML sería interesante, ya que permitiría validar los documentos antes de trabajar con ellos.

Apéndice A

DeTraS 0.4.1 User Manual

About this document

A.1. Introduction

DeTraS is a set of tools to track developers activities by registering its applications' usage. It can be also used as a personal time manager on computer, allowing to know how much time we spend performing tasks on the computer. As a lot of networking applications, it has two parts: a client (part of the application that runs on your computer) and a server (usually runs in a remote computer). This manual describes utils under client side of DeTraS (so I will use DeTraS referring to client side of DeTraS from now on). DeTraS includes these utilities:

TempusFugit. It's a tool to track your applications' usage under X11 Window.

Squealer. A tool to filter and upload your data to a machine running DeTraS server.

DeTraS applet. A GNOME applet that allows you to configure and use DeTraS' tools more easily.

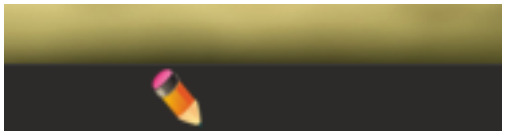

Figura A.1: DeTraS applet icon (tracking on).

A.2. Getting started

A.2.1. DeTraS installation

In order to use DeTraS, you have to download it from this page[1] and install it on your system. As this is not an installation guide, I will only give some brief steps to perform installation. There are three methods to install DeTraS on your system:

1. **Installing from a tarball**. Basically, you have to download `detras0.4.1.tar.gz` file, extract it, install packages listed on `DEPENDENCIES` file, run `autogen.sh`, `make` and `make install`.

2. **Installing from a deb package**. Download `detras_0.4.1-1_i386.deb` or `detras_0.4.1-1_amd64.deb` depending on your system architecture. Then install package double clicking on it or opening a terminal and typing `sudo dpkg -i <package>`.

3. **Installing from a repository**. This is the best option if you want to get automatically updates. To install from a repository, go to this page[2], and follow instructions under *Technical details about this PPA*. Although that page only shows information about configuring software sources on Ubuntu distros, you should be able to use this repository for almost any recent Debian based distro. Once you have configured your software sources properly, you only have to execute `sudo aptitude update` and `sudo aptitude install detras` to get DeTraS installed on your system.

A.2.2. What information tracks DeTraS?

DeTraS will store and upload this information about usage of your system:

[1] `https://launchpad.net/detras/+download`
[2] `https://launchpad.net/~edmundoa/+archive/detras-ppa`

- When you change windows focus, it will take the time, application name and its window title (you can filter windows titles before upload data).
- Time when your session become active or inactive.
- Your name (optional).
- Projects you are working on (optional).

A.2.3. What should I do if I have any question or a problem?

Try reading questions already answered on DeTraS Launchpad page[3] —see Answers section[4]— or send your own question. You have to register in Launchpad before sending any question.

If you prefer not joining Launchpad, you can write me an e-mail to `e.alvarezj@gmail.com`.

A.2.4. What should I do if I find a bug?

I really appreciate if you report any bug you found on DeTraS Launchpad page[5] —see Bugs section[6]—. You have to register in Launchpad in order to report a bug.

As I told you previously, if you prefer not joining Launchpad, you can write me an e-mail to `e.alvarezj@gmail.com`.

A.3. TempusFugit

You can interact with TempusFugit using a command line interface or DeTraS applet.

Run TempusFugit on a shell. To execute TempusFugit on a shell, you have to run `tempusfugit` command. By default, it will store all events on a file called `tempusfugit.xml` under `$HOME/.detras` folder. As you will see, TempusFugit run as a daemon, so you have to send a SIGTERM signal (Ctrl+C) to stop its execution. Be sure you send it a SIGTERM signal, because SIGKILL can break events

[3] `https://launchpad.net/detras`
[4] `https://answers.launchpad.net/detras`
[5] `https://launchpad.net/detras`
[6] `https://bugs.launchpad.net/detras`

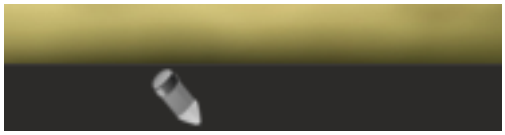

Figura A.2: DeTraS applet icon (tracking off).

file. To see more details about TempusFugit execution you can check `tempusfugit` manual page.

Run TempusFugit using DeTraS applet. See A.6 section.

A.4. Squealer

Squealer can be invoked from a shell or using DeTraS applet to upload tracked information.

- **Run Squealer on a shell**. To execute Squealer from a terminal, you must execute `squealer` command. It will apply any filter defined on preferences and will send data to configured server. Squealer will not show any output unless it encounter any problem. You can find more details on `squealer` manpage.
- **Run Squealer using DeTraS applet**. See A.6 section.

A.5. Preferences

DeTraS stores your preferences in a file called `detras.ini` under `$HOME/.detras` folder. If this file does not exist, DeTraS will get its preferences from `/usr/share/detras/detras.ini`, which contains system default preferences. Please read `detras.ini` manual page before editing these files manually.

A.6. Using DeTraS applet

A.6.1. Adding DeTraS applet to panel

To add DeTraS applet to panel, right-click on the panel and choose *Add to panel*. Then select DeTraS applet from applets' list and click on *Add* button. You should see a pencil icon in the panel that represents DeTraS applet.

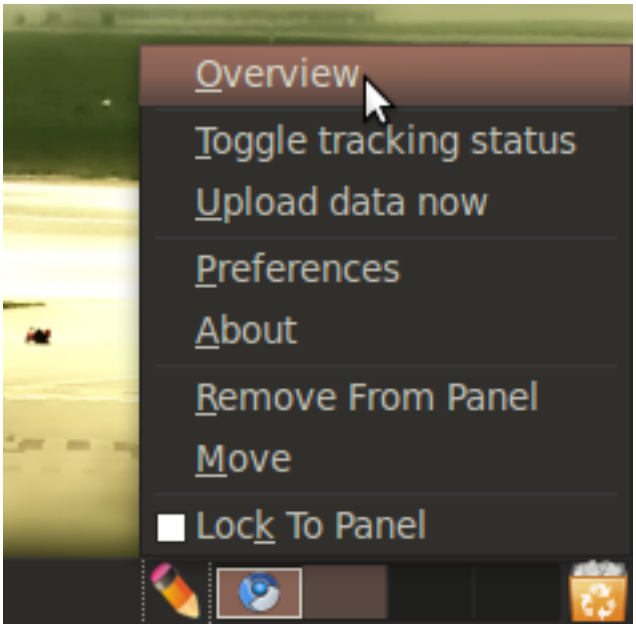

Figura A.3: DeTraS applet menu.

A.6.2. DeTraS applet menu

You can perform many actions right clicking on DeTraS applet. As you can see on figure A.3, you are able to see an overview of latest events tracked, toggle tracking status, upload data and change your preferences.

A.6.3. Work with TempusFugit

You can start or stop TempusFugit in two ways: clicking on DeTraS applet or choosing *Toggle tracking status* from DeTraS applet menu. DeTraS applet icon shows you current TempusFugit's status. A grey scale icon indicates that TempusFugit is stopped, so it is not tracking your data. A colored one indicates that TempusFugit is running.

A.6.4. Work with Squealer

To upload data tracked by TempusFugit you have to select *Upload data now* option from DeTraS applet menu. DeTraS applet will open up a window once the upload is finished or an error is found.

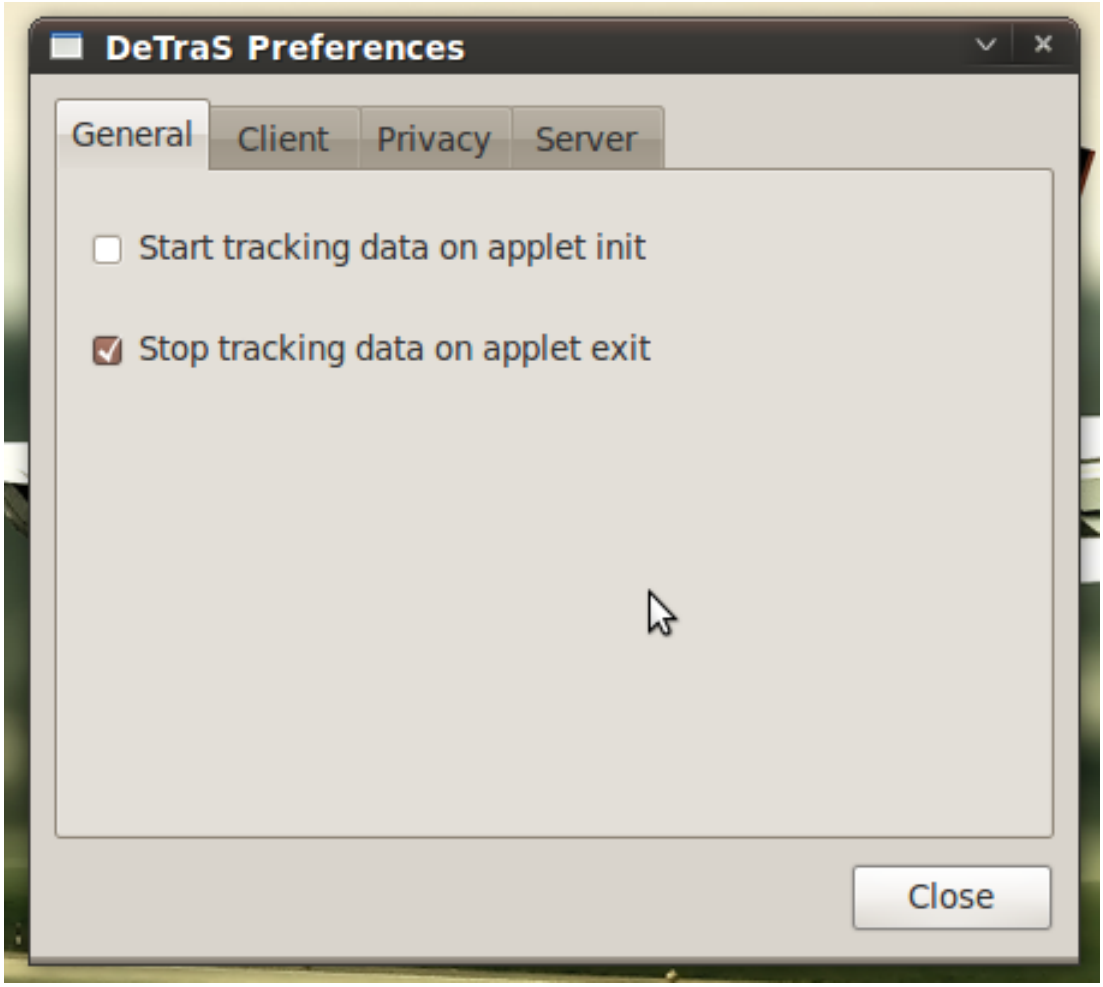

Figura A.4: DeTraS general preferences.

A.6.5. Overview window

Selecting *Overview* option on DeTraS applet menu will show you a window with latest events tracked on your system. This dialog will show you events tracked since the last time you upload your data, and you can order them by date, length, name...

A.6.6. Preferences dialog

When you click on *Preferences* item on DeTraS applet menu, you can edit DeTraS preferences in an easier way than editing `$HOME/.detras/detras.ini` file manually.

General tab

Here you can configure general preferences:

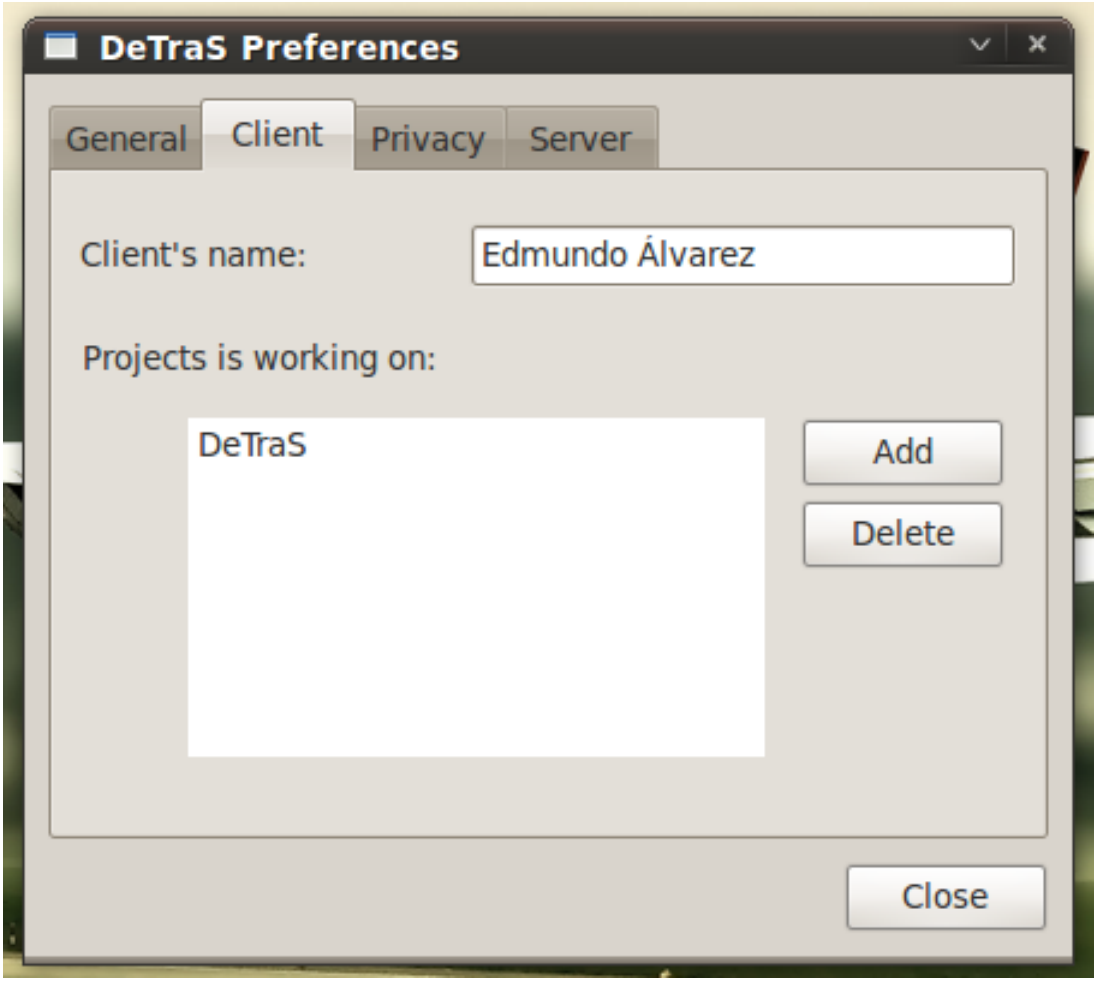

Figura A.5: DeTraS client preferences.

- **Start tracking data on applet init**. If you check this box, TempusFugit will start automatically to track your activity when the applet is loaded. (Default: disabled).
- **Stop tracking data on applet exit**. If you check this box, TempusFugit will stop tracking your activity automatically when the applet is unloaded. (Default: enabled).

Client tab

This window allows you to change client's preferences:

- **Client's name**. Type client's name on this text box.
- **Projects is working on**. Configure projects in which the client is working on.

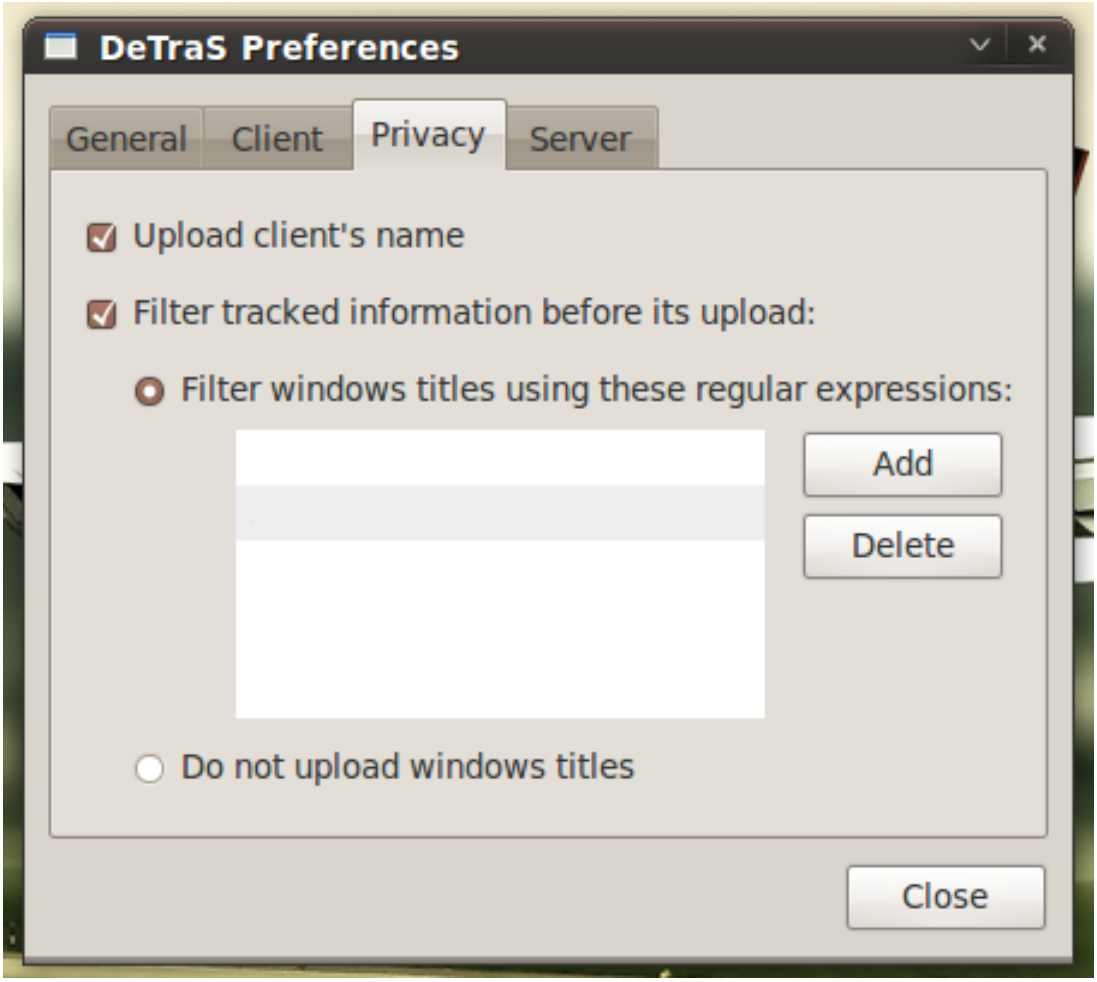

Figura A.6: DeTraS privacy preferences.

Click on *Add* button to add a new project, select a project and click *Delete* button to remove a project, or double-click on a project to edit it.

Privacy tab

This window lets you modify privacy preferences:

- **Upload client's name**. Check this box to upload client's name together with your tracked data. (Default: disabled).
- **Filter tracked information before its upload**. Check this box to enable filtering of your data. (Default: disabled).
    - **Filter windows titles using these regular expressions**. Check this button to enable filtering of windows titles. You can add regular expressions clicking

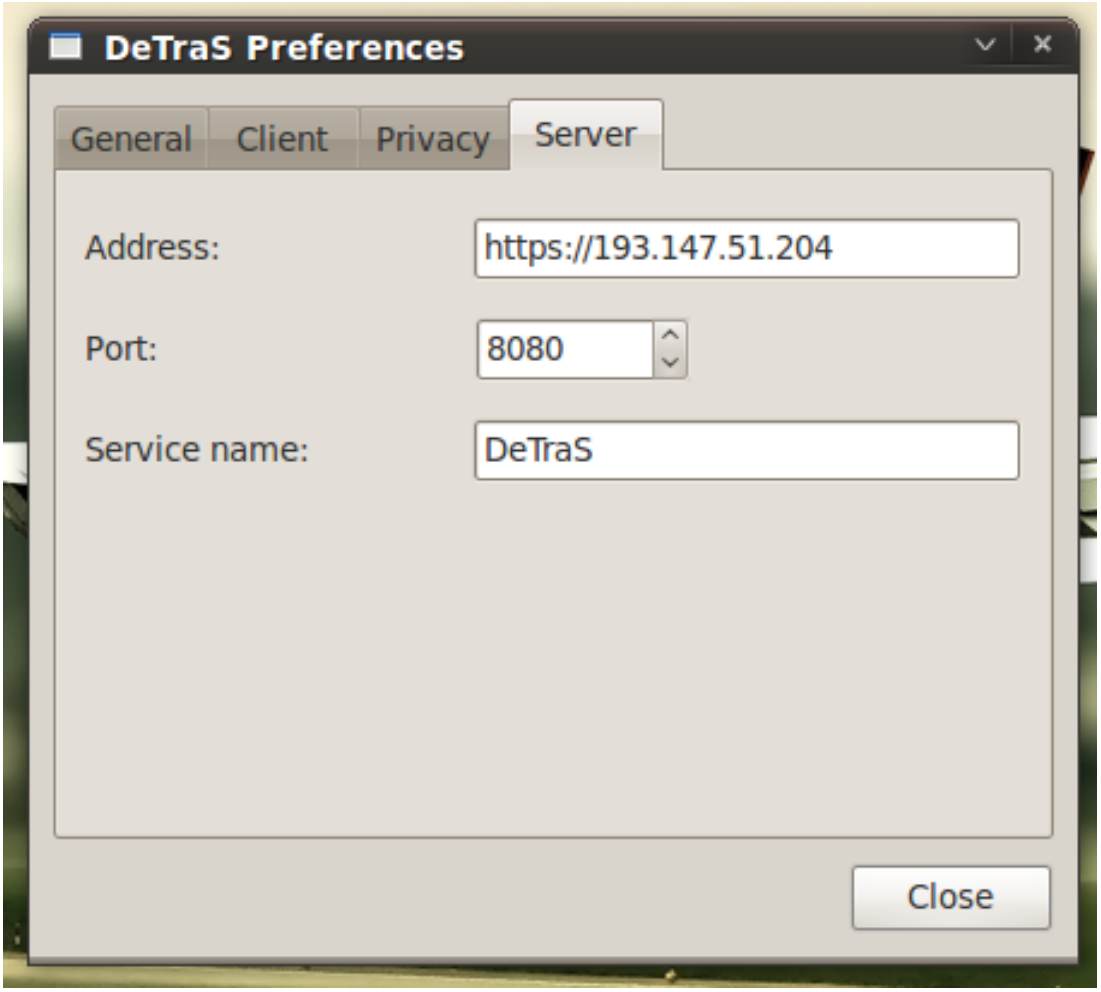

Figura A.7: DeTraS server preferences.

on *Add* button, delete one of them selecting it and clicking on *Delete* button and edit a regular expression double clicking on it. If you want help to build regular expressions, you can check this guide[7].

- **Do not upload windows titles**. Check this button to hide all information about windows titles before its uploaded.

Server tab

Configure server parameters from this window. By default, this window contains parameters of a server that is tracking data to help me doing my thesis. I really appreciate if you send your data there.

- **Address**. Type server URL. (Default: https://193.147.51.204).

[7] `http://docs.python.org/library/re.html#regular-expression-syntax`

- **Port**. Type port where the server is listening. (Default: 8080).
- **Service name**. Connection service name. Please don't change this preference unless you know what you are doing. (Default: DeTraS).

Apéndice B

Contenido del disco compacto

B.1. Licencia del código fuente

El código fuente desarrollado en este proyecto se distribuye bajo la licencia pública GPL. En el fichero `licencia.txt` dentro de la carpeta `/codigo_fuente/` del disco compacto adjunto a este proyecto se encuentra el fichero con la licencia completa utilizada. Asimismo, en todos los ficheros de código incluidos puede encontrarse el preámbulo de esta licencia:

```
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

B.2. Contenido del disco compacto

El disco compacto adjunto a este documento incluye todo el código fuente desarrollado para el proyecto, los paquetes creados para Debian, el manual de usuario, así como una versión de este documento —incluyendo sus fuentes en LaTeX—.

B.2.1. Código fuente del proyecto

El código fuente del proyecto se incluye en el directorio `/codigo/` del disco compacto. En él se encuentran todos los ficheros necesarios para su compilación y ejecución.

B.2.2. Paquetes Debian

Los paquetes Debian instalables para las arquitecturas *x86* y *amd64* se encuentran en el directorio `/debian/` del disco.

B.2.3. Manual de usuario

El manual de usuario de la versión de DeTraS incluida en el CD está disponible en la carpeta `/manual/`. Únicamente se incluye su versión en PDF, aunque puede usar las fuentes de LaTeX que se encuentran en el directorio `/codigo/doc/` para generar otro formato que sea de su agrado.

B.2.4. Memoria del proyecto

Este documento, junto a sus fuentes en LaTeX, se incluyen en el directorio `/memoria/`.

Apéndice C

Licencia de este documento

LA OBRA O LA PRESTACIÓN (SEGÚN SE DEFINEN MÁS ADELANTE) SE PROPORCIONA BAJO LOS TÉRMINOS DE ESTA LICENCIA PÚBLICA DE CREATIVE COMMONS (**CCPL O LICENCIA**). LA OBRA O LA PRESTACIÓN SE ENCUENTRA PROTEGIDA POR LA LEY ESPAÑOLA DE PROPIEDAD INTELECTUAL Y/O CUALESQUIERA OTRAS NORMAS QUE RESULTEN DE APLICACIÓN. QUEDA PROHIBIDO CUALQUIER USO DE LA OBRA O PRESTACIÓN DIFERENTE A LO AUTORIZADO BAJO ESTA LICENCIA O LO DISPUESTO EN LA LEY DE PROPIEDAD INTELECTUAL.

MEDIANTE EL EJERCICIO DE CUALQUIER DERECHO SOBRE LA OBRA O LA PRESTACIÓN, USTED ACEPTA Y CONSIENTE LAS LIMITACIONES Y OBLIGACIONES DE ESTA LICENCIA, SIN PERJUICIO DE LA NECESIDAD DE CONSENTIMIENTO EXPRESO EN CASO DE VIOLACIÓN PREVIA DE LOS TÉRMINOS DE LA MISMA. EL LICENCIADOR LE CONCEDE LOS DERECHOS CONTENIDOS EN ESTA LICENCIA, SIEMPRE QUE USTED ACEPTE LOS PRESENTES TÉRMINOS Y CONDICIONES.

1. **Definiciones**

   a) La **obra** es la creación literaria, artística o científica ofrecida bajo los términos de esta licencia.

   b) En esta licencia se considera una **prestación** cualquier interpretación, ejecución, fonograma, grabación audiovisual, emisión o transmisión, mera fotografía u otros objetos protegidos por la legislación de propiedad intelectual vigente aplicable.

c) La aplicación de esta licencia a una **colección** (definida más adelante) afectará únicamente a su estructura en cuanto forma de expresión de la selección o disposición de sus contenidos, no siendo extensiva a éstos. En este caso la colección tendrá la consideración de obra a efectos de esta licencia.

d) El **titular originario** es:

1) En el caso de una obra literaria, artística o científica, la persona natural o grupo de personas que creó la obra.
2) En el caso de una obra colectiva, la persona que la edite y divulgue bajo su nombre, salvo pacto contrario.
3) En el caso de una interpretación o ejecución, el actor, cantante, músico, o cualquier otra persona que represente, cante, lea, recite, interprete o ejecute en cualquier forma una obra.
4) En el caso de un fonograma, el productor fonográfico, es decir, la persona natural o jurídica bajo cuya iniciativa y responsabilidad se realiza por primera vez una fijación exclusivamente sonora de la ejecución de una obra o de otros sonidos.
5) En el caso de una grabación audiovisual, el productor de la grabación, es decir, la persona natural o jurídica que tenga la iniciativa y asuma la responsabilidad de las fijaciones de un plano o secuencia de imágenes, con o sin sonido.
6) En el caso de una emisión o una transmisión, la entidad de radiodifusión.
7) En el caso de una mera fotografía, aquella persona que la haya realizado.
8) En el caso de otros objetos protegidos por la legislación de propiedad intelectual vigente, la persona que ésta señale.

e) Se considerarán **obras derivadas** aquellas obras creadas a partir de la licenciada, como por ejemplo: las traducciones y adaptaciones; las revisiones, actualizaciones y anotaciones; los compendios, resúmenes y extractos; los arreglos musicales y, en general, cualesquiera transformaciones de una obra literaria, artística o científica. Para evitar la duda, si la obra consiste en una composición musical o grabación de sonidos, la sincronización temporal de

la obra con una imagen en movimiento (synching) será considerada como una obra derivada a efectos de esta licencia.

f) Tendrán la consideración de **colecciones** la recopilación de obras ajenas, de datos o de otros elementos independientes como las antologías y las bases de datos que por la selección o disposición de sus contenidos constituyan creaciones intelectuales. La mera incorporación de una obra en una colección no dará lugar a una derivada a efectos de esta licencia.

g) El **licenciador** es la persona o la entidad que ofrece la obra o prestación bajo los términos de esta licencia y le concede los derechos de explotación de la misma conforme a lo dispuesto en ella.

h) **Usted** es la persona o la entidad que ejercita los derechos concedidos mediante esta licencia y que no ha violado previamente los términos de la misma con respecto a la obra o la prestación, o que ha recibido el permiso expreso del licenciador de ejercitar los derechos concedidos mediante esta licencia a pesar de una violación anterior.

i) La **transformación** de una obra comprende su traducción, adaptación y cualquier otra modificación en su forma de la que se derive una obra diferente. La creación resultante de la transformación de una obra tendrá la consideración de obra derivada.

j) Se entiende por **reproducción** la fijación directa o indirecta, provisional o permanente, por cualquier medio y en cualquier forma, de toda la obra o la prestación o de parte de ella, que permita su comunicación o la obtención de copias.

k) Se entiende por **distribución** la puesta a disposición del público del original o de las copias de la obra o la prestación, en un soporte tangible, mediante su venta, alquiler, préstamo o de cualquier otra forma.

l) Se entiende por **comunicación pública** todo acto por el cual una pluralidad de personas, que no pertenezcan al ámbito doméstico de quien la lleva a cabo, pueda tener acceso a la obra o la prestación sin previa distribución de ejemplares a cada una de ellas. Se considera comunicación pública la puesta a disposición del público de obras o prestaciones por procedimientos alám-

bricos o inalámbricos, de tal forma que cualquier persona pueda acceder a ellas desde el lugar y en el momento que elija.

m) La **explotación** de la obra o la prestación comprende la reproducción, la distribución, la comunicación pública y, en su caso, la transformación.

n) Los **elementos de la licencia** son las características principales de la licencia según la selección efectuada por el licenciador e indicadas en el título de esta licencia: Reconocimiento, CompartirIgual.

ñ) Una **licencia equivalente** es:

1) Una versión posterior de esta licencia de Creative Commons con los mismos elementos de licencia.

2) La misma versión o una versión posterior de esta licencia de cualquier otra jurisdicción reconocida por Creative Commons con los mismos elementos de la licencia (ejemplo: Reconocimiento-CompartirIgual 3.0 Japón).

3) La misma versión o una versión posterior de la licencia de Creative Commons no adaptada a ninguna jurisdicción (Unported) con los mismos elementos de la licencia.

4) Una de las licencias compatibles que aparece en `http://creativecommons.org/compatiblelicenses` y que ha sido aprobada por Creative Commons como esencialmente equivalente a esta licencia porque, como mínimo:

a′ Contiene términos con el mismo propósito, el mismo significado y el mismo efecto que los elementos de esta licencia.

b′ Permite explícitamente que las obras derivadas de obras sujetas a ella puedan ser distribuidas mediante esta licencia, la licencia de Creative Commons no adaptada a ninguna jurisdicción (Unported) o una licencia de cualquier otra jurisdicción reconocida por Creative Commons, con sus mismos elementos de licencia.

2. **Límites de los derechos**. Nada en esta licencia pretende reducir o restringir cualesquiera límites legales de los derechos exclusivos del titular de los derechos de propiedad intelectual de acuerdo con la Ley de propiedad intelectual o cuales-

quiera otras leyes aplicables, ya sean derivados de usos legítimos, tales como la copia privada o la cita, u otras limitaciones como la resultante de la primera venta de ejemplares (agotamiento).

3. **Concesión de licencia**. Conforme a los términos y a las condiciones de esta licencia, el licenciador concede, por el plazo de protección de los derechos de propiedad intelectual y a título gratuito, una licencia de ámbito mundial no exclusiva que incluye los derechos siguientes:

   a) Derecho de reproducción, distribución y comunicación pública de la obra o la prestación.

   b) Derecho a incorporar la obra o la prestación en una o más colecciones.

   c) Derecho de reproducción, distribución y comunicación pública de la obra o la prestación lícitamente incorporada en una colección.

   d) Derecho de transformación de la obra para crear una obra derivada siempre y cuando se incluya en ésta una indicación de la transformación o modificación efectuada.

   e) Derecho de reproducción, distribución y comunicación pública de obras derivadas creadas a partir de la obra licenciada.

   f) Derecho a extraer y reutilizar la obra o la prestación de una base de datos.

   g) Para evitar cualquier duda, el titular originario:

      1) Conserva el derecho a percibir las remuneraciones o compensaciones previstas por actos de explotación de la obra o prestación, calificadas por la ley como irrenunciables e inalienables y sujetas a gestión colectiva obligatoria.

      2) Renuncia al derecho exclusivo a percibir, tanto individualmente como mediante una entidad de gestión colectiva de derechos, cualquier remuneración derivada de actos de explotación de la obra o prestación que usted realice.

Estos derechos se pueden ejercitar en todos los medios y formatos, tangibles o intangibles, conocidos en el momento de la concesión de esta licencia. Los derechos mencionados incluyen el derecho a efectuar las modificaciones que sean precisas

técnicamente para el ejercicio de los derechos en otros medios y formatos. Todos los derechos no concedidos expresamente por el licenciador quedan reservados, incluyendo, a título enunciativo pero no limitativo, los derechos morales irrenunciables reconocidos por la ley aplicable. En la medida en que el licenciador ostente derechos exclusivos previstos por la ley nacional vigente que implementa la directiva europea en materia de derecho sui generis sobre bases de datos, renuncia expresamente a dichos derechos exclusivos.

4. **Restricciones**. La concesión de derechos que supone esta licencia se encuentra sujeta y limitada a las restricciones siguientes:

   a) Usted puede reproducir, distribuir o comunicar públicamente la obra o prestación solamente bajo los términos de esta licencia y debe incluir una copia de la misma, o su Identificador Uniforme de Recurso (URI). Usted no puede ofrecer o imponer ninguna condición sobre la obra o prestación que altere o restrinja los términos de esta licencia o el ejercicio de sus derechos por parte de los concesionarios de la misma. Usted no puede sublicenciar la obra o prestación. Usted debe mantener intactos todos los avisos que se refieran a esta licencia y a la ausencia de garantías. Usted no puede reproducir, distribuir o comunicar públicamente la obra o prestación con medidas tecnológicas que controlen el acceso o el uso de una manera contraria a los términos de esta licencia. Esta sección 4.a también afecta a la obra o prestación incorporada en una colección, pero ello no implica que ésta en su conjunto quede automáticamente o deba quedar sujeta a los términos de la misma. En el caso que le sea requerido, previa comunicación del licenciador, si usted incorpora la obra en una colección y/o crea una obra derivada, deberá quitar cualquier crédito requerido en el apartado 4.c, en la medida de lo posible.

   b) Usted puede distribuir o comunicar públicamente una obra derivada en el sentido de esta licencia solamente bajo los términos de la misma u otra licencia equivalente. Si usted utiliza esta misma licencia debe incluir una copia o bien su URI, con cada obra derivada que usted distribuya o comunique públicamente. Usted no puede ofrecer o imponer ningún término respecto a la obra derivada que altere o restrinja los términos de esta licencia o el ejercicio de sus derechos por parte de los concesionarios de la misma. Us-

ted debe mantener intactos todos los avisos que se refieran a esta licencia y a la ausencia de garantías cuando distribuya o comunique públicamente la obra derivada. Usted no puede ofrecer o imponer ningún término respecto de las obras derivadas o sus transformaciones que alteren o restrinjan los términos de esta licencia o el ejercicio de sus derechos por parte de los concesionarios de la misma. Usted no puede reproducir, distribuir o comunicar públicamente la obra derivada con medidas tecnológicas que controlen el acceso o uso de la obra de una manera contraria a los términos de esta licencia. Si utiliza una licencia equivalente debe cumplir con los requisitos que ésta establezca cuando distribuya o comunique públicamente la obra derivada. Todas estas condiciones se aplican a una obra derivada en tanto que incorporada a una colección, pero no implica que ésta tenga que estar sujeta a los términos de esta licencia.

c) Si usted reproduce, distribuye o comunica públicamente la obra o la prestación, una colección que la incorpore o cualquier obra derivada, debe mantener intactos todos los avisos sobre la propiedad intelectual e indicar, de manera razonable conforme al medio o a los medios que usted esté utilizando:

1) El nombre del autor original, o el seudónimo si es el caso, así como el del titular originario, si le es facilitado.
2) El nombre de aquellas partes (por ejemplo: institución, publicación, revista) que el titular originario y/o el licenciador designen para ser reconocidos en el aviso legal, las condiciones de uso, o de cualquier otra manera razonable.
3) El título de la obra o la prestación si le es facilitado.
4) El URI, si existe, que el licenciador especifique para ser vinculado a la obra o la prestación, a menos que tal URI no se refiera al aviso legal o a la información sobre la licencia de la obra o la prestación.
5) En el caso de una obra derivada, un aviso que identifique la transformación de la obra en la obra derivada (p. ej., "traducción castellana de la obra de Autor Original,.º "guión basado en obra original de Autor Original").

Este reconocimiento debe hacerse de manera razonable. En el caso de una obra derivada o incorporación en una colección estos créditos deberán aparecer como mínimo en el mismo lugar donde se hallen los correspondientes a otros autores o titulares y de forma comparable a los mismos. Para evitar la duda, los créditos requeridos en esta sección sólo serán utilizados a efectos de atribución de la obra o la prestación en la manera especificada anteriormente. Sin un permiso previo por escrito, usted no puede afirmar ni dar a entender implícitamente ni explícitamente ninguna conexión, patrocinio o aprobación por parte del titular originario, el licenciador y/o las partes reconocidas hacia usted o hacia el uso que hace de la obra o la prestación.

d) Para evitar cualquier duda, debe hacerse notar que las restricciones anteriores (párrafos 4.a, 4.b y 4.c) no son de aplicación a aquellas partes de la obra o la prestación objeto de esta licencia que únicamente puedan ser protegidas mediante el derecho sui generis sobre bases de datos recogido por la ley nacional vigente implementando la directiva europea de bases de datos.

5. **Exoneración de responsabilidad** A MENOS QUE SE ACUERDE MUTUAMENTE ENTRE LAS PARTES, EL LICENCIADOR OFRECE LA OBRA O LA PRESTACIÓN TAL CUAL (ON AN .ᴬS-IS"BASIS) Y NO CONFIERE NINGUNA GARANTÍA DE CUALQUIER TIPO RESPECTO DE LA OBRA O LA PRESTACIÓN O DE LA PRESENCIA O AUSENCIA DE ERRORES QUE PUEDAN O NO SER DESCUBIERTOS. ALGUNAS JURISDICCIONES NO PERMITEN LA EXCLUSIÓN DE TALES GARANTÍAS, POR LO QUE TAL EXCLUSIÓN PUEDE NO SER DE APLICACIÓN A USTED.

6. **Limitación de responsabilidad**. SALVO QUE LO DISPONGA EXPRESA E IMPERATIVAMENTE LA LEY APLICABLE, EN NINGÚN CASO EL LICENCIADOR SERÁ RESPONSABLE ANTE USTED POR CUALESQUIERA DAÑOS RESULTANTES, GENERALES O ESPECIALES (INCLUIDO EL DAÑO EMERGENTE Y EL LUCRO CESANTE), FORTUITOS O CAUSALES, DIRECTOS O INDIRECTOS, PRODUCIDOS EN CONEXIÓN CON ESTA LICENCIA O EL USO DE LA OBRA O LA PRESTACIÓN, INCLUSO SI EL LICENCIADOR HUBIERA SIDO INFORMADO DE LA POSIBILIDAD DE TALES DAÑOS.

7. **Finalización de la licencia**

   a) Esta licencia y la concesión de los derechos que contiene terminarán automáticamente en caso de cualquier incumplimiento de los términos de la misma. Las personas o entidades que hayan recibido de usted obras derivadas o colecciones bajo esta licencia, sin embargo, no verán sus licencias finalizadas, siempre que tales personas o entidades se mantengan en el cumplimiento íntegro de esta licencia. Las secciones 1, 2, 5, 6, 7 y 8 permanecerán vigentes pese a cualquier finalización de esta licencia.

   b) Conforme a las condiciones y términos anteriores, la concesión de derechos de esta licencia es vigente por todo el plazo de protección de los derechos de propiedad intelectual según la ley aplicable. A pesar de lo anterior, el licenciador se reserva el derecho a divulgar o publicar la obra o la prestación en condiciones distintas a las presentes, o de retirar la obra o la prestación en cualquier momento. No obstante, ello no supondrá dar por concluida esta licencia (o cualquier otra licencia que haya sido concedida, o sea necesario ser concedida, bajo los términos de esta licencia), que continuará vigente y con efectos completos a no ser que haya finalizado conforme a lo establecido anteriormente, sin perjuicio del derecho moral de arrepentimiento en los términos reconocidos por la ley de propiedad intelectual aplicable.

8. **Miscelánea**

   a) Cada vez que usted realice cualquier tipo de explotación de la obra o la prestación, o de una colección que la incorpore, el licenciador ofrece a los terceros y sucesivos licenciatarios la concesión de derechos sobre la obra o la prestación en las mismas condiciones y términos que la licencia concedida a usted.

   b) Cada vez que usted realice cualquier tipo de explotación de una obra derivada, el licenciador ofrece a los terceros y sucesivos licenciatarios la concesión de derechos sobre la obra objeto de esta licencia en las mismas condiciones y términos que la licencia concedida a usted.

   c) Si alguna disposición de esta licencia resulta inválida o inaplicable según la Ley vigente, ello no afectará la validez o aplicabilidad del resto de los térmi-

nos de esta licencia y, sin ninguna acción adicional por cualquiera las partes de este acuerdo, tal disposición se entenderá reformada en lo estrictamente necesario para hacer que tal disposición sea válida y ejecutiva.

d) No se entenderá que existe renuncia respecto de algún término o disposición de esta licencia, ni que se consiente violación alguna de la misma, a menos que tal renuncia o consentimiento figure por escrito y lleve la firma de la parte que renuncie o consienta.

e) Esta licencia constituye el acuerdo pleno entre las partes con respecto a la obra o la prestación objeto de la licencia. No caben interpretaciones, acuerdos o condiciones con respecto a la obra o la prestación que no se encuentren expresamente especificados en la presente licencia. El licenciador no estará obligado por ninguna disposición complementaria que pueda aparecer en cualquier comunicación que le haga llegar usted. Esta licencia no se puede modificar sin el mutuo acuerdo por escrito entre el licenciador y usted.

Bibliografía

[1] GARCÍA CAMPOS, C. "DeTraS: sistema de seguimiento de actividad de desarrollo". Director: Gregorio Robles Martínez. Universidad Rey Juan Carlos de Madrid, 205/2006.

[2] PRESSMAN, R.S. *Ingeniería del Software, un enfoque práctico*. 5a ed. Madrid: McGraw-Hill, 2002.

[3] SOMMERVILLE, I. *Software Engineering*. 8a ed. Harlow, England: Addison-Wesley, 2007.

[4] LÓPEZ, J.E.; DOLADO COSÍN, J.J. "Estimación del Esfuerzo Software: Factores vinculados a la aplicación a desarrollar". *Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos*. Vol. 2, No. 1, 2008.

[5] GARZÁS PARRAS, J. *Introducción a la estimación* [en línea]. <http://jgarzas.googlepages.com/Jgarzas_Estimacion_Putnam.pdf> [Consulta: noviembre de 2009].

[6] KEYES, J. *Software Engineering Handbook*. 1a ed. Florida: CRC Press, 2003.

[7] MORENO CAPUCHINO, A.M. *COCOMO II* [en línea]. <http://trevinca.ei.uvigo.es/~cfajardo/Nueva_carpeta/presentaciones/cocomo2k.pdf> [Consulta: diciembre de 2009].

[8] AMOR, J.J.; ROBLES, G.; GONZÁLEZ–BARAHONA, J.M. *Effort Estimation by Characterizing Developer Activity*. 2006.

[9] *Administración de tiempo* [en línea]. Wikilibros. <http://es.wikibooks.org/wiki/Administraci%C3%B3n_de_tiempo/Versi%C3%B3n_para_imprimir> [Consulta: noviembre de 2009].

[10] *The Free Software Definition* [en línea]. Free Software Foundation. <http://www.gnu.org/philosophy/free-sw.html> [Consulta: diciembre de 2009].

[11] KERNIGHAN, B.W; RITCHIE, D.M. *The C programming language*. 2a ed. [s.l.]: Prentice Hall, 1988.

[12] *Lenguaje de programación C* [en línea]. Wikipedia. <http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_C> [Consulta: febrero de 2010].

[13] *GLib Reference Manual* [en línea]. GNOME Project. <http://library.gnome.org/devel/glib/2.22/> [Consulta: diciembre de 2009 – mayo de 2010].

[14] *GObject Reference Manual* [en línea]. GNOME Project. <http://library.gnome.org/devel/gobject/2.22/> [Consulta: diciembre de 2009 – mayo de 2010].

[15] *D-Bus GLib bindings - Reference Manual* [en línea]. GNOME Project. <http://library.gnome.org/devel/dbus-glib/unstable/> [Consulta: diciembre de 2009 – mayo de 2010].

[16] *Extensible Markup Language* [en línea]. Wikipedia. <http://es.wikipedia.org/wiki/XML> [Consulta: junio de 2010].

[17] GONZALEZ DUQUE, R. *Python para todos* [en línea]. <http://mundogeek.net/tutorial-python/> [Consulta: noviembre de 2009].

[18] *About Python* [en línea]. Python Software Foundation. <http://python.org/about/> [Consulta: junio de 2010].

[19] *Django (web framework)* [en línea]. Wikipedia. <http://en.wikipedia.org/wiki/Django_(web_framework)> [Consulta: junio de 2010].

[20] *HTML* [en línea]. Wikipedia. <http://es.wikipedia.org/wiki/Html> [Consulta: junio de 2010].

[21] FLANAGAN, D. *JavaScript, The Definitive Guide*. 5a ed. Sebastopol, CA: O'Reilly Media, 2006.

[22] *Dojo toolkit* [en línea]. Wikipedia. <http://es.wikipedia.org/wiki/Dojo_toolkit> [Consulta: junio de 2010].

[23] *Cascading Style Sheets* [en línea]. Wikipedia. <http://en.wikipedia.org/wiki/Cascading_Style_Sheets> [Consulta: junio de 2010].

[24] *Google Chart Tools / Image Charts (aka Chart API)* [en línea]. Google. <http://code.google.com/apis/chart/image_charts.html> [Consulta: mayo – junio de 2010].

[25] *Python Google Chart* [en línea]. <http://pygooglechart.slowchop.com/> [Consulta: mayo – junio de 2010].

[26] *SQLite* [en línea]. Wikipedia. <http://es.wikipedia.org/wiki/SQLite> [Consulta: junio de 2010].

[27] *About SQLite* [en línea]. SQLite. <http://www.sqlite.org/about.html> [Consulta: junio de 2010].

[28] *GTK+* [en línea]. Wikipedia. <http://es.wikipedia.org/wiki/GTK%2B> [Consulta: junio de 2010].

[29] *PyGTK* [en línea]. Wikipedia. <http://en.wikipedia.org/wiki/PyGTK> [Consulta: junio de 2010].

[30] *Glade* [en línea]. Wikipedia. <http://es.wikipedia.org/wiki/Glade> [Consulta: junio de 2010].

[31] *Bazaar (software)* [en línea]. Wikipedia. <http://es.wikipedia.org/wiki/Bazaar_(software)> [Consulta: junio de 2010].

[32] *Launchpad (website)* [en línea]. Wikipedia. <http://en.wikipedia.org/wiki/Launchpad_(website)> [Consulta: junio de 2010].