UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Gºen Ingeniería en Informática

Trabajo final del GºIng.Informática:

Monitor multiplataforma de la actividad de un proyecto

Presentado por David Blanco Alonso
en febrero de 2016
Tutor Carlos López Nozal

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Gºen Ingeniería en Informática

D. Carlos López Nozal, profesor del departamento de Ingeniería Ingeniería Civil, área de Área de Lenguajes y Sistemas Informáticos

Expone:

Que el alumno D. David Blanco Alonso, con DNI 71290575, ha realizado el Trabajo final del GºIng.Informática titulado: Monitor multiplataforma de la actividad de un proyecto.

y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual, Se autoriza su presentación y defensa.

En Burgos a 3 de febrero de 2016

Carlos López Nozal

Índice de contenido

Índice de ilustraciones

Índice de tablas

Resumen

Actualmente existen varias plataformas de desarrollo colaborativo para proyectos software que funcionan con filosofías que heredan de los entornos de desarrollo ágiles, adoptar estrategias de desarrollo incremental, equipos de trabajo auto gestionados y solapamiento en las fases de desarrollo. Estas plataformas proporcionan un alojamiento en la nube para el proyecto además de control de versiones y sistemas de seguimiento de issues.

Una de estas plataformas es GitHub [1] que utiliza un control de versiones basado en git y un sistema de seguimiento de issues propio además funcionar de forma similar a una red social ya que por defecto todos los proyectos son públicos, los usuarios pueden colaborar o subscribirse a los proyectos que les resulten interesantes.

Debido a la cantidad de proyectos existentes en la plataforma, actualmente 31 millones según la propia GitHub, siempre existen proyectos que se han estancado o abandonado temporalmente, proyectos que intentan resolver el mismo problema con distintos enfoques o lenguajes de programación, Por ello los usuarios que se interesan por un proyecto utilizan la información que proporciona GitHub para todos los proyectos para valorar y diferenciar unos de otros, pero esta información no siempre resulta suficiente para conocer el verdadero estado de un proyecto.

Este proyecto se basa en la utilización de los datos que proporciona GitHub sobre un proyecto para mostrar una serie de métricas basadas en una selección de las descritas en la tesis sPACE: Software Project Assessment in the Course of Evolution, desarrollada por Jacek Ratzinguer [2]. Mediante estas métricas se pretende tener una visión más profunda de la evolución y el estado de un proyecto, así como de una visión más o menos superficial de la forma de trabajo.

Abstract

Currently there are several collaborative development platforms for software projects that work with philosophies that inherit from agile development environments, adopt strategies of incremental development, self-managed work teams and overlapping phases of development. These platforms provide cloud hosting for the project in addition to version control and tracking systems issues.

One of these platforms is GitHub [1] that uses a version control based on git and segimiento system of own issues also operate similarly to a social network since by default all projects are public, users can collaborate or subscribe to the projects that they find interesting.

Due to the number of existing projects in the platform, currently 31 million according to GitHub itself , there are always projects that have stalled or temporarily abandoned projects that attempt to solve the same problem with different approaches or programming languages, therefore users an interest in a project using information provided by GitHub for all projects to assess and differentiate from each other , but this information is not always enough to know the true state of a project.

This project is based on the use of the data provided on a GitHub project to show a series of metrics based on a selection from those described in the thesis Space: Software Project Assessment in the Course of Evolution, developed by Jacek Ratzinger [2]. Using these metrics is to have a deeper insight into the evolution and status of a project, as well as a more or less superficial vision of the way they work.

I - Introducción

A lo largo de este documento vamos a exponer los aspectos mas relevantes durante el desarrollo del trabajo de fin de grado descrito anteriormente, estructurándolo según la siguiente estructura [3]:

- Introducción: Breve descripción del propósito del proyecto, del contenido de la memoria y de la estructura de la misma.
- Objetivos del proyecto: Descripcion de los objetivos que se pretenden alcanzar con este proyecto.
- Conceptos teóricos: Explicación de los distintos apartados teóricos que se deben comprender para la satisfactoria realización del proyecto.
- Técnicas y herramientas: Enumeración de los aspectos mas destacados de las técnicas metodológicas y las herramientas de desarrollo que se han utilizado en la realización del proyecto.
- Aspectos relevantes del desarrollo del proyecto: Explicación de los aspectos mas destacados de las fases de análisis, diseño e implementación.
- Trabajos relacionados: Descripcion de trabajos y proyectos realizados anteriormente en el campo en el que se realiza el proyecto.
- Conclusiones y lineas de trabajo futuras: Resumen de las conclusiones obtenidas en la realización del proyecto y posibles ideas para la mejora y continuidad del proyecto.

II - Objetivos del proyecto

El objetivo principal de este proyecto es implementar un software capaz de proporcionar una información mas detallada, clara y precisa de los datos que se pueden obtener directamente desde la plataforma GitHub [1] para permitir conocer de forma rápida el estado y la evolución de un proyecto software alojado en ella.

Para ello se debe implementar un conjunto de métricas adecuado extraído de la tesis sPACE: Software Project Assessment in the Course of Evolution, desarrollada por Jacek Ratzinguer [2] y de versiones propias de las mostradas por la propia GitHub.

También se pretende que se puedan añadir nuevas plataformas (Bitbucket [4], Trello [5], etc.) a la aplicación para poder obtener los datos de los proyectos alojados en ellas de forma sencilla.

III - Conceptos teóricos

Para la correcta realización de este proyecto debemos utilizar diversos conceptos teóricos de varios campos diversos como pueden ser los patrones de diseño, las metodologías ágiles etc.

1. Patrones de diseño [6]

Lo primero que debemos de hacer antes de trabajar con los patrones de diseño es entender que es un patrón de diseño para poder utilizarlos de forma adecuada, lo primero es entender las definiciones que existen sobre los patrones de diseño una de ellas es "Los patrones de diseño son soluciones para problemas típicos y recurrentes que nos podemos encontrar a la hora de desarro-

llar una aplicación" otra definición es "Un patrón es una solución general, fruto de la experiencia, a un problema general que puede adaptarse a un problema concreto". Estas definiciones nos dicen que existen patrones ya definidos para problemas de carácter general que ya han sido más que probados para solucionar problemas concretos por otros programadores, por lo que si entendemos que problema general se adapta a nuestro problema concreto podemos utilizar el patrón que soluciona ese problema general.

El siguiente paso es definir cuál es nuestro problema y compararlo con los problemas de carácter general que solucionan los distintos patrones para encontrar que patrón o patrones nos permiten solucionar nuestro problema concreto, para la realización del proyecto hemos encontrado los patrones de fabrica abstracta, fachada y singleton.

- Patrón fachada: El problema que pretendemos solucionar al emplear este patrón es que realizando un estudio de las distintas plataformas existentes para el desarrollo de proyectos, cada una tiene una estructura propia a la hora de guardar la información sobre los commits o las issues. También a la hora de realizar las peticiones cada plataforma como es lógico tiene una dirección de red a la que hay que realizar la petición, es decir para obtener las issues de GitHub debemos realizar la petición sobre la dirección "https://api.github.com/repos/{usuario}/{repositorio}/issues" en cambio para obtenerlas de Bitbucket debemos utilizar la dirección "https://bitbucket.org/api/1.0/repositories/{usuario}/{repositorio}/issues". Este patrón fachada soluciona este problema ocultando al usuario las diferencias existentes haciendo que para él sea lo mismo pedir los datos de una plataforma o de otra.
- Fabrica abstracta: El problema que se pretende solucionar con este patrón es la creación de la fachada necesaria para realizar las operaciones de forma correcta sobre la plataforma seleccionada, cada fachada trabaja sobre una plataforma. El usuario decide sobre que plataforma desea trabajar y la fábrica se encarga de crearla para evitar que el usuario deba conocer aspectos del software que puede que escapen a su comprensión.
- Singleton: Este patrón soluciona los problemas que puedan surgir si se crean varias fábricas abstractas o fachadas para trabajar con la plataforma que ha seleccionado el usuario.

Todos estos patrones y problemas surgen en previsión de que se pretenda añadir implementaciones para trabajar con otras plataformas de desarrollo de software aparte de GitHub que es la única que está actualmente implementada.

2. *Métricas de medición*

Debemos entender que métricas pueden ser atractivas para el usuario a la hora de comprender y poder valorar el estado y la evolución de un proyecto software y si es posible obtener la información necesaria de la plataforma para realizar esa métrica.

A continuación vamos a definir las métricas utilizadas agrupadas por categorías.

De las métricas descritas en la tesis sPACE: Software Project Assessment in the Course of Evolution, desarrollada por Jacek Ratzinguer [2] vamos a seleccionar las que podemos realizar ya sea porque no podemos obtener el numero de líneas modificadas en cada cambio o commit ni el numero de bits modificado.

2.1. Equipo

Para todo proyecto de desarrollo software uno de los aspectos más importantes es el trabajo en equipo. Estas métricas nos permiten conocer el número de autores que están trabajando en el proyecto y una estimación de la actividad que están realizando.

- CambioPorAutor: Numero de cambios realizados por cada autor.

Nombre	CambioPorAutor
Propósito	¿Cuántos commits ha realizado cada usuario?
Medición, fórmula	CPA cambio por autor
Interpretación	CPA > 0 mejor valores altos
Tipo de escala	Absoluta
Tipo de medida	CPA contador
Fuente de Medición	Repositorio GitHub de un proyecto

Tabla 1: Descripción de la métrica CambioPorAutor según el formato ISO 9126.

- IssuesPorAutor: Numero de issues creadas por cada autor.

Nombre	IssuesPorAutor
Propósito	¿Cuántas issues ha realizado cada usuario?
Medición, fórmula	IPA issues por usuario
Interpretación	IPA >= 0 mejor valores altos
Tipo de escala	Absoluta
Tipo de medida	IPA contador
Fuente de Medición	Repositorio GitHub de un proyecto

Tabla 2: Descripción de la métrica IssuesPorAutor según el formato ISO 9126.

- ContadorAutor: Número de autores trabajando, normalizado sobre el número total de cambios.

Nombre	ContadorAutor
Propósito	¿Cuál es la relación entre el número de desarrolladores y el número de cambios del proyecto?
Medición, fórmula	CA = NA (Numero autores) / NC (Numero cambios)
Interpretación	CA > 0 mejor valores bajos
Tipo de escala	Absoluta
Tipo de medida	NA contador, NC contador
Fuente de Medición	Repositorio GitHub de un proyecto

Tabla 3: Descripción de la métrica ContadorAutor según el formato ISO 9126.

2.2. Proceso de orientación

Esta categoría recoge las métricas que muestran el comportamiento del equipo de trabajo con respecto a las tareas o issues que se crean en el proyecto, cuánto tiempo tardan en cerrarlas, porcentaje de cierre etc. Con respecto a las métricas descritas en la tesis sPACE: Software Project Assessment in the Course of Evolution, desarrollada por Jacek Ratzinguer hemos tenido que adaptar varias métricas ya que en la plataforma GitHub no se puede dar prioridad a las tareas que se declaran.

- NumeroIssues: Numero de issues creadas en el proyecto.

Nombre	NumeroIssues
Propósito	¿Cuántas issues se han creado en el repositorio?
Medición, fórmula	NI número de issues
Interpretación	NI >= 0 mejor valores bajos
Tipo de escala	Absoluta
Tipo de medida	NI contador
Fuente de Medición	Repositorio GitHub de un proyecto

Tabla 4: Descripción de la métrica NumeroIssues según el formato ISO 9126.

- NumeroIssuesCerradas: Numero de issues que se han cerrado.

Nombre	NumeroIssuesCerradas
Propósito	¿Cuántas issues se han cerrado en el repositorio?
Medición, fórmula	NIC número de issues cerradas
Interpretación	NIC >= 0 mejor valores altos
Tipo de escala	Absoluta
Tipo de medida	NIC contador
Fuente de Medición	Repositorio GitHub de un proyecto

Tabla 5: Descripción de la métrica NumeroIssuesCerradas según el formato ISO 9126.

- MediaDiasCierre: Media de días que se tarda en cerrar una issue.

Nombre	MediaDiasCierre
Propósito	¿Cuánto se tarda de media en cerrar una issue?
Medición, fórmula	MDC = D (suma de los días) / NIC (numero de issues cerradas)
Interpretación	MDC >= 0 mejor valores bajos
Tipo de escala	Absoluta
Tipo de medida	D contador, NIC contador
Fuente de Medición	Repositorio GitHub de un proyecto

Tabla 6: Descripción de la métrica MediaDiasCierre según el formato ISO 9126.

- PorcentajeIssuesCerradas: Porcentaje de issues que se han cerrado sobre las que se han creado.

Nombre	PorcentajeIssuesCerradas
Propósito	¿Proporción de issues cerradas en el repositorio en función de las creadas?
Medición, fórmula	PIC = NIC (Número de issues cerradas) * 100 / NI (Número de issues)
Interpretación	0 <= PIC <= 100 mejor valores altos
Tipo de escala	Ratio
Tipo de medida	NIC contador, NI contador
Fuente de Medición	Repositorio GitHub de un proyecto

Tabla 7: Descripción de la métrica PorcentajeIssuesCerradas según el formato ISO 9126.

- NumeroCambiosSinMensaje: Numero de commits sin mensaje.

Nombre	NumeroCambiosSinMensaje
Propósito	¿Cuántos cambios se han realizado sin mensaje?
Medición, fórmula	NCSM número de cambios sin mensaje
Interpretación	NCSM >= 0 mejor valores bajos
Tipo de escala	Absoluta
Tipo de medida	NCSM contador
Fuente de Medición	Repositorio GitHub de un proyecto

Tabla 8: Descripción de la métrica NumeroCambiosSinMensaje según el formato ISO 9126.

- ContadorTareas: Número de tareas, normalizada sobre el número total de cambios.

Nombre	ContadorTareas
Propósito	¿Cuál es el volumen medio de trabajo de las tareas?
Medición, fórmula	CT = NT (Numero de tareas) / NTC (Numero total de cambios)
Interpretación	CT >= 0 mejor valores intermedios
Tipo de escala	Ratio
Tipo de medida	NT contador, NTC contador
Fuente de Medición	Repositorio GitHub de un proyecto

Tabla 9: Descripción de la métrica ContadorTareas según el formato ISO 9126.

2.3. Restricciones temporales

Las siguientes métricas son las relacionadas con las restricciones temporales. Para todo proyecto de desarrollo de software es interesante saber el tiempo que se aplica en las distintas acciones para poder prever cómo van a afectar futuros cambios o acciones en la evolución del proyecto y su coste en el mismo.

- MediaDiasCambio: Numero de días de media entre cambios.

Nombre	MediaDiasCambio
Propósito	¿Cuantos días de media pasan entre cambios?
Medición, fórmula	MDC = D (Días desde el primer al último cambio) / NTC (Número total de cambios)
Interpretación	MDC >= 0 mejor valores bajos
Tipo de escala	Absoluta
Tipo de medida	D contador, NTC contador
Fuente de Medición	Repositorio GitHub de un proyecto

Tabla 10: Descripción de la métrica MediaDiasCambio según el formato ISO 9126.

- DiasPrimerUltimoCommit: Días que han pasado desde que se realizo el primer commit hasta que se realizo el ultimo.

Nombre	DiasPrimerUltimoCommit
Propósito	¿Cuantos días han pasado entre el primer y el último commit?
Medición, fórmula	DPUC días pasados
Interpretación	DPUC >= 0 mejor valores altos
Tipo de escala	Absoluta
Tipo de medida	DPUC contador
Fuente de Medición	Repositorio GitHub de un proyecto

Tabla 11: Descripción de la métrica DiasPrimerUltimoCommit según el formato ISO 9126.

- UltimaModificacion: Fecha en la que se realizo el ultimo commit.

Nombre	UltimaModificacion
Propósito	¿Cuando se realizo el último cambio en el repositorio?
Medición, fórmula	UM fecha
Interpretación	
Tipo de escala	Absoluta
Tipo de medida	UM fecha
Fuente de Medición	Repositorio GitHub de un proyecto

Tabla 12: Descripción de la métrica UltimaModificacion según el formato ISO 9126.

- CommitPorDia: Numero de commits que se han realizado cada día de la semana durante todo el proyecto.

Nombre	CommitPorDia
Propósito	¿Cuántos commits se han realizado cada día de la semana?
Medición, fórmula	CPD commits por día
Interpretación	CPD >= 0 mejor valores altos
Tipo de escala	Absoluta
Tipo de medida	CPD contador
Fuente de Medición	Repositorio GitHub de un proyecto

Tabla 13: Descripción de la métrica CommitPorDia según el formato ISO 9126.

- CommitPorMes: Numero de commits que se han realizado cada mes durante todo el proyecto.

Nombre	CommitPorMes
Propósito	¿Cuántos commits se han realizado cada mes?
Medición, fórmula	CPM commits por mes
Interpretación	CPM >= 0 mejor valores altos
Tipo de escala	Absoluta
Tipo de medida	CPM contador
Fuente de Medición	Repositorio GitHub de un proyecto

Tabla 14: Descripción de la métrica CommitPorMes según el formato ISO 9126.

- RelacionMesPico: Mes en que más cambios se han realizado.

Nombre	RelacionMesPico
Propósito	¿Cuál es el mes en que más cambios se han realizado?
Medición, fórmula	RMP mes en el que más cambios se han realizado
Interpretación	
Tipo de escala	Nominal
Tipo de medida	RMP mes
Fuente de Medición	Repositorio GitHub de un proyecto

Tabla 15: Descripción de la métrica RelacionMesPico según el formato ISO 9126.

- ContadorCambiosPico: Número de cambios en el mes que más se han realizado, normalizado sobre el número total de cambios.

Nombre	ContadorCambiosPico
Propósito	¿Cuál es la proporción de trabajo realizado en el mes con mayor número de cambios?
Medición, fórmula	CCP = NCMP (Número de cambios en el Mes Pico) / NTC (Número total de cambios)
Interpretación	0 <= CCP <= 1 Mejor valores intermedios
Tipo de escala	Ratio
Tipo de medida	NCMP contador, NTC contador
Fuente de Medición	Repositorio GitHub de un proyecto

Tabla 16: Descripción de la métrica ContadorCambiosPico según el formato ISO 9126.

- RatioActividadCambio: Número de cambios relativos al número de meses.

Nombre	RatioActividadCambio
Propósito	¿Cuál es el número medio de cambios por mes?
Medición, fórmula	RAC = (NTC = Número total de cambios) / NM (Número de meses)
Interpretación	RAC > 0 Mejor valores intermedios
Tipo de escala	Ratio
Tipo de medida	NTC contador, NM contador
Fuente de Medición	Repositorio GitHub de un proyecto

Tabla 17: Descripción de la métrica RatioActividadCambio según el formato ISO 9126.

2.4. Importancia del proyecto

Estas métricas nos da una idea de la importancia del proyecto al mostrar el número de usuarios de la plataforma que han marcado el proyecto como favorito.

- NumeroFavoritos: Número de usuarios que han marcado como favorito el proyecto.

Nombre	NumeroFavoritos
Propósito	¿Cuántos usuarios han declarado como favorito el proyecto?
Medición, fórmula	NF Favoritos
Interpretación	NF >= 0 mejor valores altos
Tipo de escala	Absoluta
Tipo de medida	NF contador
Fuente de Medición	Repositorio GitHub de un proyecto

Tabla 18: Descripción de la métrica NumeroFavoritos según el formato ISO 9126.

2.5. Metodologías ágiles

Las metodologías ágiles heredan del manifiesto ágil publicado en febrero de 2001 que aboga por dar mayor valor a los individuos y sus interacciones sobre los procesos y las herramientas, valora el software funcional por encima de la documentación, la colaboración con el cliente sobre la negociación contractual y la respuesta al cambio por encima de la planificación total.

Sobre estos principios recogidos en el manifiesto ágil surgieron varias metodologías basadas en el desarrollo iterativo e incremental, en las cuales los requisitos y soluciones evolucionan en cada iteración mediante la colaboración de equipos auto-organizados y multidisciplinarios. El desarrollo se realiza en cortos lapsos de tiempo, iteraciones, que permiten adaptar los requisitos a nuevos eventos externos o internos al desarrollo. Cada iteración tiene su planificación, análisis de requisitos, diseño, implementación, revisión y documentación. El objetivo de cada iteración es obtener un prototipo sin errores que cumpla los requisitos que se solicitaban en esa misma iteración. Al final de cada iteración el equipo vuelve a evaluar las prioridades del proyecto.

Uno de los ejemplos de metodologías ágiles es Scrum [7]:

Scrum divide a los participantes en el desarrollo del producto por medio de varios roles cada uno con unas características, deberes y objetivos:

- Product Owner: Este rol representa la voz del cliente. Se asegura de que el equipo trabaje de forma adecuada desde la perspectiva del negocio.
- ScrumMaster: Su principal objetivo es solucionar los problemas que impiden alcanzar el objetivo del sprint. Es el que se asegura de que el proceso scrum se utiliza como es debido.
- Equipo de desarrollo: Es el responsable de entregar el producto. Formado por entre 3 y 9 personas con habilidades transversales necesarias para realizar el trabajo.

Scrum también tiene un conjunto de reuniones utilizadas para dirigir el desarrollo:

- Daily scrum: Se realizan cada día de un sprint, sirven para poner de manifiesto las tareas que está realizando cada miembro del equipo y declarar si se ha encontrado algún problema para alcanzar los objetivos.
- Scrum de Scrum: Estas reuniones se realizan después del daily scrum entre varios equipos de trabajo y se centran en áreas de solapamiento e integración.
- Sprint Planning Meeting: Esta reunión se realiza antes de cada sprint y sirve para planificarlo.

- Sprint Review Meeting: Esta reunión se realiza al final de cada sprint para revisar que trabajo se ha completado y presentar una demo o prototipo a los interesados.
- Sprint Retrospective: Esta reunión se realiza también al finalizar el sprint y en ella todos los miembros del equipo exponen sus impresiones sobre el sprint, con el objetivo de mejorar el proceso.

Un sprint es el periodo de tiempo en el que se realiza el trabajo, es recomendable que su duración sea constante y definida por el equipo en función de sus experiencias previas, al final de cada sprint se presentan los resultados alcanzados, principalmente en forma de prototipo que se presenta al cliente comprobar que el rumbo del proyecto es el deseado.

2.6. GitHub

GitHub [1], [8] es una plataforma de desarrollo colaborativo que permite alojar proyectos, proporcionando además de un facilitar un espacio de alojamiento en la red proporciona un sistema de control de versiones basado en git. GitHub provee a sus usuarios de una plataforma en la que alojar sus proyectos, con un control de versiones integrado, crea una página web para cada proyecto en la que se pueden ver varios gráficos que proporcionan información referente a los cambios realizados en el proyecto, commits al mes, commits al día, visitantes, líneas de código cambiadas al día, forks o bifurcaciones del proyecto, etc.

Una de las principales características de GitHub es que tiene aspectos que coinciden con los de una red social, permite a desarrolladores de distintas partes del mundo ponerse en contacto y buscar proyectos en desarrollo que coincidan con sus ideas o buscar apoyos para desarrollar sus propias ideas. Como bien indica uno de sus eslóganes "Build software better, together." esta es una de sus características más importantes y a la que quieren dar mayor importancia.

Vocabulario de GitHub

Para entender mejor algunas descripciones de las métricas con las que trabaja actualmente la aplicación vamos a proceder a indicar a que se refiere cada palabra:

- Commit: Se refiere a los cambios realizados en el repositorio del proyecto como modificar líneas de código, eliminar o crear archivos.
- Issue: Son las tareas que crean los desarrolladores para indicar que acciones deben realizar como corregir bugs, añadir características, etc.
- Fork: Son bifurcaciones del proyecto con las que se pretende trabajar en nuevas funcionalidades sin entorpecer la línea de trabajo general del proyecto.

IV - Técnicas y herramientas

1. GitHub Java API (org.eclipse.egit.github.core)

GitHub Java API [9] es una biblioteca java para la comunicación con la API de GitHub. Está enfocada a permitir un uso completo de la API v3 GitHub, La biblioteca actualmente está siendo utilizada por el conector GitHub Mylyn para trabajar con las issues, pull request, gists y repositorios desde dentro de eclipse.

Esta biblioteca incorpora la posibilidad de autenticación básica y OAuth2 token, que permiten un número más elevado de peticiones a GitHub (5000/hora) que sin autenticar (60/hora). También controla la paginación de las peticiones rest para aquellas peticiones cuya respuesta contenga demasiados datos como para devolverla en una sola respuesta.

2. Alternativas estudiadas

Al principio del desarrollo comencé a implementar una biblioteca propia para realizar las peticiones a la API de GitHub para solicitar la información de las issues, commits y repositorios. Pero ante el numero tan reducido de peticiones sin autenticar la conexión, decidí buscar una forma de realizar dicha conexión autenticándola y fue cuando encontré la biblioteca GitHub Java API, que resolvía el problema de la autenticación y de la paginación. Por lo que opte por sustituir una biblioteca por otra.

3. JavaHelp

JavaHelp [10] es una expansión de Java que permite una fácil programación de ventanas de ayuda en aplicaciones java. La especificación y API permiten a los desarrolladores personalizar y ampliar el sistema de ayuda para adaptarse al estilo y necesidades de sus aplicaciones. El sistema JavaHelp ha sido diseñado para funcionar especialmente bien en una variedad de entornos de red. El sistema JavaHelp es independiente de la plataforma y funciona en todos los navegadores que soportan la plataforma Java.

Voy a utilizar expansión de Java para crear una ventana de ayuda en la aplicación, para resolver algunas dudas y mostrar las métricas creadas.

4. JFreeChart

JFreeChart [11] es una biblioteca gratuita de Java que facilita a los desarrolladores mostrar gráficos de calidad en sus aplicaciones, proporcionando diversas características:

- Una API consistente y bien documentada, soportando una amplia gama de tipos de gráficos.
- Un diseño flexible que es fácil de extender tanto del lado del servidor como del lado del cliente de aplicaciones.
- Soporte para muchos tipos de salida, incluyendo Swing y componentes JavaFX, archivos de imagen (incluyendo PNG y JPEG) y formatos de archivos de gráficos vectoriales (incluyendo PDF, EPS y SVG).

Esta biblioteca se va a utilizar para generar gráficos que mejoren la comprensión de algunas métricas.

5. Eclipse

Eclipse [12] proporciona entornos de desarrollo integrados y plataformas de casi todos los idiomas y arquitecturas. Además de un IDE Eclipse proporciona una amplia gama de herramientas y complementos de fácil instalación. Eclipse también es una comunidad que se encarga de aumentar el número de herramientas y plugins existentes.

El IDE de java de Eclipse será usado para el desarrollo del proyecto.

6. ObjectAid

ObjectAid [13] es una herramienta de visualización de código ágil y ligero para el IDE de Eclipse. Muestra el código fuente de Java y bibliotecas en los diagramas de clases UML y la secuencia en directo que se actualizan automáticamente a medida que cambia de código.

Voy a usar este plugin de Eclipse para crear los diagramas de clase y de secuencia que servirán para documentar la aplicación.

7. JFormDesigner

JFormDesigner [14] es un plugin de eclipse que habilita una interfaz en el propio eclipse para facilitar la creación de interfaces de usuario con Java™ Swing.

He utilizado este plugin para la creación de toda la interfaz gráfica de usuario.

8. JUnit

JUnit [15] es un conjunto de bibliotecas creadas por Erich Gamma y Kent Beck que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java.

Esta biblioteca la utilizare para poder realizar un test que compruebe la correcta recepción y manipulación de los datos recibidos.

9. PMD

PMD es un analizador de código. PMD incluye un conjunto de reglas incorporadas y apoya la capacidad de escribir reglas personalizadas. Por norma general los problemas reportados por PMD no son errores en sí, sino fragmentos de código ineficiente.

PMD será utilizado para realizar revisiones al código y encontrar fragmentos de código duplicados.

10. Zotero

Zotero [16] es un gestor bibliográfico de código abierto, que permite instalarlo como complemento del navegador mozilla firefox o como aplicación independiente para Windows, Mac y GNU/Linux. Permite la integración con Microsoft Word y OpenOffice Writer por medio de la instalación de sendos plugins.

Este gestor va a ser utilizado para llevar un control de las referencias bibliográficas utilizadas para la realización de la documentación del proyecto.

11. OpenOffice

OpenOffice [17] es una suite ofimática de código abierto competidora directa de Microsoft Office, que incluye versiones en código abierto de todos los componentes de este como un procesador de texto, OpenOffice Writer.

Voy a utilizar el OpenOffice Writer para redactar la documentación del proyecto.

12. GitHub [1]

Como esta explicado en el apartado anterior, conceptos teóricos, GitHub es una plataforma de desarrollo colaborativo que proporciona un control de versiones y un alojamiento el desarrollo de proyectos software.

Voy a utilizar GitHub para alojar el proyecto y usar el control de versiones que proporciona.

13. Framework

El manejo de las métricas se implementara tomando como base la solución propuesta en “Soporte de Métricas con Independencia del Lenguaje para la Inferencia de Refactorizaciones” [18].

Tomare este framework como base para la gestión de las métricas, pero aplicare algunas modificaciones.

14. Gson

Gson [19] es una biblioteca de código abierto que permite la serialización y deserialización de objetos java a y desde JSON. Algunas ventajas de Gson son:

- Proporcionar sencillos métodos toJson() y fromJson() para convertir objetos Java a JSON y viceversa.
- Permitir objetos preexistentes no modificables ser convertidos desde y hacia JSON.
- Amplio soporte de los tipos genéricos de Java.
- Permitir las representaciones de los objetos personalizados.
- Da apoyo a los objetos arbitrariamente complejos (con jerarquías de herencia de profundidad y un amplio uso de los tipos genéricos).

Esta biblioteca se usa para procesar las peticiones de la biblioteca GitHub Java API.

V - Aspectos relevantes del desarrollo del proyecto

Los aspectos relevantes del desarrollo van a ser descritos como los principales hitos alcanzados durante la realización del proyecto, describiendo los pasos seguidos y los componentes utilizados y estudiados en cada hito.

1. Lectura de datos desde la plataforma

Al inicio del desarrollo se comenzó a crear una biblioteca propia que mediante peticiones rest a las plataformas de GitHub [1] y Bitbucket [4] que recibía datos de los repositorios almacenados en la plataforma y los guardaba para su estudio mediante métricas.

El uso de esta biblioteca propia tenía varios problemas como la autenticación del usuario que realiza la petición rest para obtener una tasa mayor de peticiones a la plataforma y la paginación de la respuesta para obtener todos los datos en las respuestas de gran tamaño.

Durante la búsqueda de información de cómo realizar la autenticación en la plataforma GitHub por medio de una autenticación básica, es decir por medio de un nombre de usuario y una contraseña asociada al mismo, se encontró la biblioteca GitHub Java API (org.eclipse.egit.github.core) [9] que resolvía los dos problemas que presentaba la biblioteca propia por lo que decidí utilizar esta nueva biblioteca encontrada sustituyendo la biblioteca propia.

Realice una búsqueda posterior para encontrar una biblioteca similar para la plataforma Bitbucket pero no hubo ningún resultado. Por lo que decidí centrar el proyecto únicamente en la plataforma GitHub.

2. Implementación del framework de métricas

Una vez leída la información de un repositorio es el momento de realizar los cálculos para obtener los valores de cada métrica aplicable al proyecto.

Para ello tomando como base el framework propuesto en “Soporte de Métricas con Indepen-

dencia del Lenguaje para la Inferencia de Refactorizaciones" [18] se crea un framework para manejar todas las métricas que se decida implementar.

El framework propuesto proporciona algunas funcionalidades que no son necesarias para el planteamiento del proyecto, por lo que voy a modificarlo para únicamente usarlo como controlador de todas las métricas y los valores obtenidos.

El framework al final se encarga de proporcionar una interfaz común a todas las métricas, sin tener en cuenta el tipo de datos que maneja ni los cálculos que realiza cada métrica, además recoge todas las métricas y sus valores obtenidos en un único objeto para que sea más fácil trabajar con todas las métricas en conjunto.

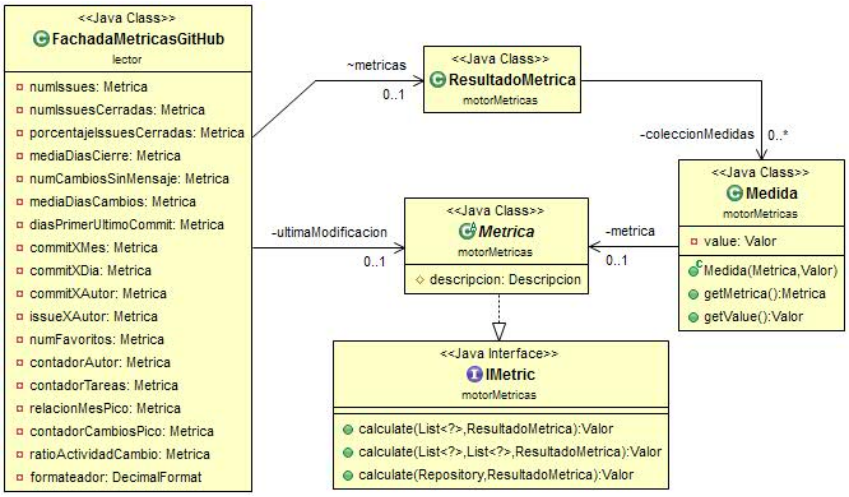


Ilustración 1: Diagrama de una parte del framework implementado

Cada métrica recibe el objeto resultadoMetricas donde se van guardando todas las métricas y sus valores obtenidos una vez calculados. La fachada es la que sabe que métricas de las creadas en el motor se pueden aplicar a los datos recibidos de esa plataforma.

3. Mostrar resultados

Una vez calculados los valores de cada métrica estos se deben mostrar al usuario, pero esto debe hacerse de la forma más visual posible para mejorar su compresión, esto lleva al uso de gráficos para mostrar los resultados alcanzados.

Por medio de la biblioteca JFreeChart [11] se crean varios gráficos para las métricas mas propensas a ser mostradas por medio de ellos y se muestran en un panel con pestañas en el que el usuario puede ir alternando para que la pantalla de resultados no se sature.

Los gráficos realizados muestran la información de las métricas PorcentajeIssuesCerradas, CommitPorDia, CommitPorMes, CambioPorAutor, IssuesPorAutor. El gráfico que muestra la información de la métrica PorcentajeIssuesCerradas es un gráfico de tarta y el resto son gráficos de barras.

Además de los gráficos también se muestran en formato texto todas las métricas con los valores obtenidos para cada una.

4. Guardado y lectura de informes

Con los resultados de cada métrica para un repositorio resulta interesante la opción de guardar estos resultados en un informe que sea posible leer sin necesidad de volver a solicitar la información del mismo a la plataforma y la realización de los cálculos para obtener los mismos resulta-

dos que ya se tienen.

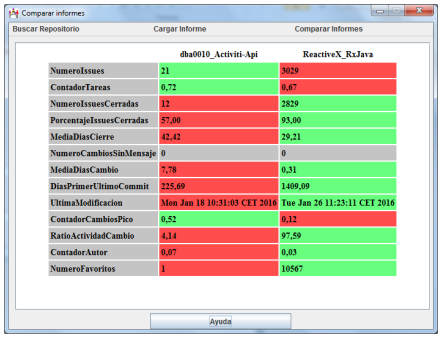
Esta funcionalidad se ha realzado añadiendo un botón a la pantalla resultados que muestra un cuadro de selección de ficheros realizado con la clase JFileChosser que permite seleccionar la ubicación del informe guardado, este informe se genera en formato texto y se guarda con una extensión txt.

Una vez guardado se puede leer un informe y obtener la misma pantalla de resultados que si volviéramos a realizar todas las peticiones de información y los cálculos de todas las métricas, lo que ahorra un gran número de peticiones a la plataforma y de tiempo si se trata de un repositorio grande.

5. Comparación de informes

Con los informes guardados una funcionalidad atractiva era la comparación de los valores alcanzados en cada métrica por dos repositorios.

Se valoro que métricas podían ser propensas de compararse y se realizo una tabla con las mismas en las que mostraba los valores alcanzados por cada repositorio aplicando un fondo verde para marcar que repositorio tiene un valor más optimo para esa métrica y un fondo rojo para los menos óptimos, si coinciden los valores no se aplica ningún fondo.



	dha0010_Activiti-Api	ReactiveX_RxJava
NumeroIssues	21	3029
ContadorTareas	0,72	0,67
NumeroIssuesCerradas	12	2829
PorcentajeIssuesCerradas	57,00	93,00
MediaDiasCierre	42,42	29,21
NumeroCambiosSinMensaje	0	0
MediaDiasCambio	7,78	0,31
DiasPrimerUltimoCommit	225,69	1409,09
UltimaModificacion	Mon Jan 18 10:31:03 CET 2016	Tue Jan 26 11:23:11 CET 2016
ContadorCambiosPico	0,52	0,12
RatioActividadCambio	4,14	97,59
ContadorAutor	0,07	0,03
NumeroFavoritos	1	10567

Ilustración 2: Muestra de la comparación de dos informes

VI - Trabajos relacionados

Existen varios proyectos que muestran estadísticas del desarrollo de un proyecto software y la propia plataforma GitHub [1] proporciona algunas estadísticas de sus proyectos alojados.

1. GitStats

GitStats [20] es un generador de estadísticas para repositorios git. Examina el repositorio y muestra estadísticas de la historio del mismo.

Muestra información de :

- Estadísticas generales: archivos totales, líneas, commits, autores.
- Actividad: commits por hora del día, por día de la semana, por hora de la semana, por mes, por mes y año, por año.
- Autores: lista de autores (nombre, commits(%), fecha del primer commit, fecha del ultimo commit, edad), autor del mes, autor del año.

- Archivos: total de archivos por fecha, extensiones.
- Líneas: líneas de código por fecha.

Actualmente su salida es únicamente en formato HTML.

2. StatCVS

StatCVS [21] recibe información de repositorios CVS y genera varias tablas y gráficos describiendo el desarrollo del proyecto.

Algunos datos que muestra:

- línea de tiempo para las líneas de código.
- Actividad por hora.
- Actividad por autor.
- Actividad de cada autor por modulo.
- Estadísticas por directorio.
- Total de archivos.
- Media de tamaño.
- Árbol del repositorio con contador de archivos y lineas de código.

3. StatSVN

StatSVN [22] muestra la información que muestra StatCVS pero para repositorios Subversion generando tablas y gráficos para mostrarlos.

4. Métricas de Evolución del Software en Plastic SCM [23]

Este es un proyecto final de carrera de Ingeniería informática de la universidad de Burgos que trata de monitorizar la evolución de software por medio de un conjunto de métricas a partir de información obtenida en los sistemas de configuración software.

Este trabajo anterior utiliza un conjunto de métricas muy parecido al que he creado para realizar mi trabajo y ha servido como base para realizar algunas mejoras a mi propio proyecto.

5. GitHub

GitHub [1] muestra información de cada repositorio alojado en forma de gráficos.

- Commits y líneas de código añadidas y eliminadas por fecha y por autor.
- Visitantes del repositorio y archivos visitados.
- Commit por mes y commits por día cada semana.
- Líneas de código añadidas frente a líneas de código borradas cada día.
- Commits realizados cada día de la semana y cada hora.
- Branches del repositorio.
- Forks creados del repositorio.

6. Comparativa de métricas

Voy a proceder a comparar en una tabla la información mostrada en los trabajos relacionados expuestos anteriormente con la mostrada en el proyecto creado.

Información	Proyecto propio	GitStats	StatCVS y StatSVN	Plastic SCM	GitHub
Tamaño del proyecto	No	Si	Si	Si	Si
Equipo	Si	Si	Si	Si	Si
Proceso de orientación	Si	No	No	Si	Si
Complejidad de la solución existente	No	No	Si	Si	No
Dificultad del problema	No	No	No	Si	No
Restricciones temporales	Si	Si	Si	Si	Si
Descubrimiento de defectos	No	No	No	Si	No
Clasificación por categorías	No	No	Si	Si	No
Importancia del proyecto	Si	No	No	No	Si

Tabla 19: Comparativa de la información proporcionada.

VII - Conclusiones y líneas de trabajo futuras

Paso a detallar las conclusiones obtenidas de la realización del proyecto y de posibles formas de continuar trabajando en el mismo.

1. Conclusiones

Durante la realización de este proyecto he alcanzado varios objetivos técnicos propuestos al inicio de su desarrollo:

- Comprensión y utilización de técnicas de metodología ágil para el desarrollo de proyectos software y para comprender el estado de un proyecto.
- Creación de un conjunto de métricas que aporte información del estado y del modelo de desarrollo seguido en un repositorio.
- Desarrollo de una aplicación con un entorno gráfico que permite seleccionar un repositorio alojado en la plataforma GitHub [1] y muestra los valores alcanzados en el conjunto de métricas creado.
- Desarrollar un software que permita añadir métricas o plataformas paras trabajar de forma sencilla mediante la aplicación de patrones de diseño.

Durante la realización de este proyecto se han comprendido bastante mejor las ventajas del uso de plataformas de desarrollo de proyectos que permiten alojar proyectos, proporcionando un control de versiones, control de tareas y disponibilidad en red. Estas ventajas se pueden apreciar mas cuando se forma parte de un equipo de desarrollo, pero siendo una sola persona se pueden comenzar a apreciar como mejoran el control del desarrollo del proyecto.

2. Líneas de trabajo futuras

Tras la finalización de este proyecto existen varias líneas trabajo por las que continuar mejorando la experiencia de la aplicación obtenida, mediante la implementación de nuevas características o la mejora de funcionalidades ya existentes.

- Añadir más plataformas a la ya existente de GitHub para obtener sus repositorios y permitir que se pueda aplicar el conjunto de métricas creado a los mismos, se puede realizar creando una biblioteca propia o buscando una ya creada que realice la conexión y las peticiones rest.
- Varias de las métricas que no se han podido implementar son las relacionadas con las líneas de código modificadas en cada commit por dificultades a la hora de obtener dicha información de la plataforma, para obtenerla es necesaria realizar una petición extra por cada commit existente en el repositorio, lo que implica que para repositorios con gran numero de commits se alcance el numero de peticiones permitido por la plataforma solo obteniendo las líneas de código.
- Uno de los aspectos más importantes del proyecto es presentar los resultados de la forma más sencilla y comprensiva para el usuario por lo que nunca hay que dejar de trabajar para mejorar la interfaz de usuario, mejorando su diseño y añadiendo mejores componentes para mostrar los resultados.
- Otra de las posibles mejoras a introducir es la de permitir al usuario seleccionar que métricas de las disponibles quiere utilizar, esto permitiría un mayor control al usuario de la información proporcionada por la aplicación, obviando información que no necesita y centrándose en la que prioriza.

VIII - REFERENCIAS

1. Bibliografía

[1] «GitHub · Where software is built». [En línea]. Disponible en: https://github.com/. [Accedido: 01-feb-2016].
[2] Jacek Ratzinger, «sPACE Software Project Assessment in the Course of Evolution». .
[3] F. J. García Peñalvo, J. M. Maudes Raedo, M. G. Piattini Velthuis, J. R. García-Bermejo Giner, y M. N. Moreno García, «Proyecto de Final de Carrera en la Ingeniería Técnica en Informática: Guía de Realización y Documentación». 1999.
[4] «Bitbucket — The Git solution for professional teams». [En línea]. Disponible en: https://bitbucket.org/. [Accedido: 01-feb-2016].
[5] «Trello». [En línea]. Disponible en: https://trello.com/. [Accedido: 03-feb-2016].
[6] Erich Gamma, *Patrones de diseño*. .
[7] «Scrum - Wikipedia, la enciclopedia libre». [En línea]. Disponible en: https://es.wikipedia.org/wiki/Scrum. [Accedido: 29-ene-2016].
[8] «GitHub - Wikipedia, la enciclopedia libre». [En línea]. Disponible en: https://es.wikipedia.org/wiki/GitHub. [Accedido: 29-ene-2016].
[9] «egit-github/org.eclipse.egit.github.core at master · eclipse/egit-github». [En línea]. Disponible en: https://github.com/eclipse/egit-github/tree/master/org.eclipse.egit.github.core. [Accedido: 29-ene-2016].
[10] «JavaHelp — Project Kenai». [En línea]. Disponible en: https://javahelp.java.net/. [Accedido: 30-ene-2016].
[11] «JFreeChart». [En línea]. Disponible en: http://www.jfree.org/jfreechart/. [Accedido: 30-ene-2016].

[12] «Eclipse - The Eclipse Foundation open source community website.» [En línea]. Disponible en: https://eclipse.org/. [Accedido: 30-ene-2016].
[13] «ObjectAid UML Explorer - Home». [En línea]. Disponible en: http://www.objectaid.com/. [Accedido: 03-feb-2016].
[14] «FormDev - JFormDesigner - Java/Swing GUI Designer». [En línea]. Disponible en: http://www.formdev.com/. [Accedido: 29-ene-2016].
[15] «JUnit - About». [En línea]. Disponible en: http://junit.org/. [Accedido: 30-ene-2016].
[16] «Zotero | Home». [En línea]. Disponible en: https://www.zotero.org/. [Accedido: 30-ene-2016].
[17] «Apache OpenOffice - Official Site - The Free and Open Productivity Suite». [En línea]. Disponible en: https://www.openoffice.org/. [Accedido: 01-feb-2016].
[18] R. M. Sánchez, Y. C. González-Carvajal, y C. L. Nozal, «Soporte de Métricas con Independencia del Lenguaje para la Inferencia de Refactorizaciones», en *Actas de las X Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2005), September 14-16, 2005, Granada, Spain*, 2005, pp. 59–66.
[19] «google/gson». [En línea]. Disponible en: https://github.com/google/gson. [Accedido: 30-ene-2016].
[20] «GitStats - git history statistics generator». [En línea]. Disponible en: http://gitstats.sourceforge.net/. [Accedido: 01-feb-2016].
[21] «StatCVS - Repository Statistics - Introduction». [En línea]. Disponible en: http://statcvs.sourceforge.net/. [Accedido: 01-feb-2016].
[22] «StatSVN - Repository Statistics - Introduction». [En línea]. Disponible en: http://statsvn.org/. [Accedido: 01-feb-2016].
[23] Irene Barbero Tera y Estrella Resa Camarero, «Métricas de Evolución del Software en Plastic SCM». 2011.

Impreso en Burgos el miércoles, 3 de febrero de 2016