UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

GII_O_MA_19.07 Comparador de métricas de evolución en repositorios Software

Presentado por Joaquín García Molina
Universidad de Burgos
13 de enero de 2022
Tutor: Carlos López Nozal

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. Carlos López Nozal, profesor del departamento de nombre departamento, área de nombre área.

Expone:

Que el alumno D. Joaquín García Molina, con DNI 76441581-T, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado GII_O_MA_19.07 Comparador de métricas de evolución en repositorios Software.

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, 13 de enero de 2022

Vº. Bº. del Tutor:

D. Carlos López Nozal

Resumen

En este primer apartado se hace una **breve** presentación del tema que se aborda en el proyecto.

Descriptores

Palabras separadas por comas que identifiquen el contenido del proyecto Ej: servidor web, buscador de vuelos, android ...

Abstract

A **brief** presentation of the topic addressed in the project.

Keywords

keywords separated by commas.

Índice general

Índice de figuras

Índice de tablas

Introducción

El desarrollo de software es una actividad que puede ser enormemente compleja al poder desarrollar proyectos de gran envergadura que impliquen a muchas personas y entidades trabajando con diferentes herramientas y con variados patrones de organicación [2]. Esta complejidad existe a nivel técnico ya que se requiere que se cumplan los requisitos establecidos funcionales pero es necesario que también se cumpla con los no funcionales como pueden ser la seguridad, la posibilidad de actualización, la escalabilidad de la arquitectura elegida, los tiempos de carga, etc. Además, esta complejidad existe también a nivel organizativo, es necesario que los jefes de proyecto sean capaces de organizar a los equipos y el trabajo a realizar de forma que se optimice tanto el tiempo como los recursos económicos disponibles.

Para poder salvar esta complejidad y que los jefes de proyecto sean capaces de llevar a cabo su tarea de optimización de los recursos se han creado diferentes modelos que permiten definir las actividades y tareas realizadas en los proyectos de forma organizada y lo más sencilla posible. Entre estos modelos destaca ***Unified Process (UP)*** [2] donde se definen las siguientes tareas en un proyecto:

- Captura de requisitos.
- Análisis
- Diseño
- Implementación
- Prueba

Estas diferentes tareas o fases de realizan de forma iterativa e incremental, es decir, tras la fase de prueba se comprueba que no existen nuevos requisitos y se repite todo el proceso. Cada una de estas iteraciones ha de resultar en un entregable o artefacto a ser posible funcional.

Este desarrollo iterativo incremental está también reflejado en otras metodologías de trabajo como son las de desarrollo ágil como Scrum, Lean o eXtreme Programming.

Para poder llevar a cabo este desarrollo iterativo y de forma colaborativa, ya que como se ha indicado los proyectos de desarrollo de software implican normalmente a más de un desarrollador, es necesario disponer de herramientas que permitan tanto guardar el código como permitir compartirlo de forma sencilla con el resto del equipo o equipos. Estas herramientas son los repositorios de software que son espacios centralizados donde se almacena, organiza, mantiene y difunde información digital, habitualmente archivos informáticos, que pueden contener trabajos científicos, conjuntos de datos o software [1].

Las herramientas de control de repositorios o forjas de proyectos software han evolucionado con los años y tienen muchas más funcionalidades además del control de los propios archivos y se centran en fomentar el desarrollo colaborativo y la interacción entre desarrolladores. Entre dichas funcionalidades podemos nombrar el control de versiones, el control de los archivos de forma colaborativa, almacenándose tanto los propios archivos como las interacciones entre los miembros del equipo que los manipulan, sistemas de revisión de calidad, sistemas de control de incidencias (o ïssues") o sistemas de integración y despliegue continuo denominados CICD (Çontinuous delivery - Continuous deployment"). Entre estas herramientas podemos destacar en la actualidad por ser las más usadas: GitHub [2], GitLab [3] o Bitbucket [4]) aunque existen otras como SourceForge [5].

Estas herramientas están en continua evolución desarrollando nuevas funcionalidades para mejorar la experiencia de los desarrolladores y los gestores de proyectos y permiten integración con terceros para ofrecer aquellas características que aún no ofrecen. En el orden de las metodologías ágiles las diferentes plataformas están avanzando mucho ofreciendo por ejemplo

[1] https://es.wikipedia.org/
[2] https://github.com/
[3] https://about.gitlab.com/
[4] https://bitbucket.org/
[5] https://sourceforge.net/

GitLab el módulo GitLab Issues [6]) o GitHub GitHub Issues [7] (mejorado con ZenHub [8]).

Estas herramientas permiten generar una enorme cantidad de información de los proyectos y del proceso de desarrollo. En la actualidad uno de los aspectos del desarrollo de software que más interés despierta y en el que se realizan más avances es en la gestión de esta información ya que, cuanto más se optimice dicha gestión de la información, antes se pueden detectar los fallos en el proceso de desarrollo para optimizarlo. Este es el campo de trabajo de los jefes de proyecto, el correcto control sobre el proceso de desarrollo y el producto creado. Es por esto crucial que exista una manera de medir si se está realizando correctamente dicho proceso o no. Para ello existen las control y de predicción [4]. Las primeras se refieren al proceso de desarrollo, y las segundas al producto, en el presente TFG nos centraremos fundamentalmente en las primeras.

Es claro que el resultado de un proyecto dependerá del proceso de desarrollo seguido y su calidad. Esto es explicado, por ejemplo, por Sommerville en *Ingeniería de software* [4] y que cuanto mejor sea este proceso mejores resultados se obtendrán a la finalización de los proyectos.

Este presente TFG pretende profundizar en este punto, mejorar la calidad de los procesos de desarrollo. Pretende hacerlo realizando una nueva iteración sobre ***Evolution Metrics Gauge***, un software para calcular métricas de control [9] sobre distintos repositorios. Dicho software ha sido desarrollado en un TFG previo titulado ***Evolution Metrics Gauge - Comparador de métricas de evolución en repositorios software*** [6]. Este software consiste en una aplicación Web escrita en lenguaje Java que toma como entrada un conjunto de repositorios públicos o privados de GitLab y calcula métricas de evolución que permiten comparar los proyectos. En esta nueva iteración se pretende extender la funcionalidad a repositorios de GitHib, añadir otras métricas e implementar diferentes mejoras.

[6] https://docs.gitlab.com/ee/user/project/issues/

[7] https://docs.github.com/en/issues/tracking-your-work-with-issues/about-issues

[8] https://www.zenhub.com/

[9] También llamadas métricas de proceso o métricas de evolución

1.1. Estructura de la memoria

La presente memoria tiene la siguiente estructura [10]:

Introducción. Introducción. Estructura de la memoria y anexos.
Objetivos del proyecto. Objetivos que busca alcanzar el proyecto.
Conceptos teóricos. Definiciones de los conceptos empleados en el proyecto.
Técnicas y herramientas. Técnicas y herramientas utilizadas durante el desarrollo del proyecto.
Aspectos relevantes del desarrollo. Aspectos destacables durante el proceso de desarrollo del proyecto.
Trabajos relacionados y *debt process*. Desarrollos relacionados y deuda técnica asociada a proceso.
Conclusiones y líneas de trabajo futuras. Conclusiones tras la realización del proyecto y posibilidades de mejora o expansión.

Se incluyen también los siguientes anexos (TODO -> Pendientes de definir):

Plan del proyecto software. Planificación temporal y estudio de la viabilidad del proyecto.
Especificación de requisitos del software. Análisis de los requisitos.
Especificación de diseño. Diseño de los datos, diseño procedimental y diseño arquitectónico.
Manual del programador. Aspectos relevantes del código fuente.
Manual de usuario. Manual de uso para usuarios que utilicen la aplicación.

[10] Se parte de lap lantilla LaTex proporcionada en https://github.com/ubutfgm/plantillaLatex

Objetivos del proyecto

A continuación se van a detallar los objetivos perseguidos con la realización del proyecto así como los de la aplicación Web sobra la que se trabaja,

2.1. Objetivos generales

El objetivo principal del presente TFG es realizar mejoras y extender la funcionalidad que tiene el software ***Evolution Metrics Gauge***, un software para calcular métricas de control [11] sobre distintos repositorios. En esta nueva iteración se pretende:

- Permitir leer repositorios de GitHub además de GitLab.
- Introducir nuevas métricas a las ya evaluadas. TODO -> definir cuáles
- Realizar pruebas con repositorios de GitLab y GitHub simultáneamente.
- Realizar diferentes mejoras en las interfaces de la aplicación para mejorar la experiencia de usuario.
- Corrección de errores.
- Mejorar el paquete de tests de la aplicación para mejorar su cobertura.

[11] También llamadas métricas de proceso o métricas de evolución

2.2. Objetivos de la aplicación y funcionalidad previa

A continuación se enumeran los objetivos iniciales de la aplicación ya desarrollada y cómo se han desarrollado: [6]

- Se obtienen medidas de métricas de evolución de uno o varios proyectos alojados en repositorios de GitLab.
- Las métricas que se calculan de un repositorio son algunas de las especificadas en la tesis titulada "*sPACE: Software Project Assessment in the Course of Evolution*" [3] y adaptadas a los repositorios software:
  - Número total de incidencias (*issues*)
  - Cambios (*commits*) por incidencia
  - Porcentaje de incidencias cerrados
  - Media de días en cerrar una incidencia
  - Media de días entre cambios
  - Días entre primer y último cambio
  - Rango de actividad de cambios por mes
  - Porcentaje de pico de cambios
- Se permite comparar con otros proyectos de la misma naturaleza. Para ello se establecen unos valores umbrales por cada métrica basados en el cálculo de los cuartiles Q1 y Q3. Además, estos valores se calculan dinámicamente y se almacenan en perfiles de configuración de métricas.
- Se permite la posibilidad de almacenar de manera persistente estos perfiles de configuiración de métricas para permitir comparaciones futuras (TODO -> comprobar)
- También se permite almacenar de forma persistente las métricas obtenidas de los repositorios para su posterior consulta o tratamiento. Esto permite comparar nuevos proyectos con proyectos de los que ya se han calculado sus métricas. Esta funcionalidad se basa en la exportación e importación de los valores de métricas obtenidos por la aplicación en diferentes análisis usando archivos CSV.

2.3. Objetivos técnicos

Además de mantener todos los objetivos técnicos previos fijados para el desarrollo de la aplicación en la primera iteración, se busca profundizar en ellos y mejorar el desarrollo actual. Para ello se se proponen los siguientes objetivos:

- Mejora de la cobertura de pruebas automáticas del proyecto.
- Mejora del tratamiento de errores para evitar bloqueos de la aplicación.
- Mejora del almacenamiento y visualización de errores para facilitar su corrección de cara a posteriores iteraciones.
- Actualización y puesta a punto del sistema de integración y despliegue continuo.
- Pruebas la aplicación con ejemplos reales y utilizando técnicas avanzadas, como entrada de datos de test en ficheros con formato tabulado tipo CSV (*comma separated values*) también para métricas obtenidas de repositorios de GitHub.

2.4. Objetivos técnicos previos

Este apartado recoge los requisitos técnicos del proyecto existente [6]:

- Diseño de la aplicación de manera que se puedan extender con nuevas métricas con el menor coste de mantenimiento posible. Para ello, se aplica un diseño basado en frameworks y en patrones de diseño [1].
- El diseño de la aplicación facilita la extensión a otras plataformas de desarrollo colaborativo como GitHub o Bitbucket.
- Aplicación del *frameworks 'modelo-vista-controlador'* para separar la lógica de la aplicación y la interfaz de usuario.
- Creación una batería de pruebas automáticas con cobertura por encima del 90 % en los subsistemas de lógica de la aplicación.
- Utilización una plataforma de desarrollo colaborativo que incluya un sistema de control de versiones, un sistema de seguimiento de incidencias y que permita una comunicación fluida entre el tutor y el alumno.
- Utilización un sistema de integración y despliegue continuo.
- Correcta gestión de errores definiendo excepciones de biblioteca y registrando eventos de error e información en ficheros de *log*.
- Aplicar nuevas estructuras del lenguaje Java para el desarrollo, como son expresiones lambda.

- Utilización de sistemas que aseguren la calidad continua del código que permitan evaluar la deuda técnica del proyecto.
- Pruebas la aplicación con ejemplos reales y utilizando técnicas avanzadas, como entrada de datos de test en ficheros con formato tabulado tipo CSV (*comma separated values*).

Conceptos teóricos

En aquellos proyectos que necesiten para su comprensión y desarrollo de unos conceptos teóricos de una determinada materia o de un determinado dominio de conocimiento, debe existir un apartado que sintetice dichos conceptos.

Algunos conceptos teóricos de LaTeX[12].

3.1. Secciones

Las secciones se incluyen con el comando section.

Subsecciones

Además de secciones tenemos subsecciones.

Subsubsecciones

Y subsecciones.

3.2. Referencias

Las referencias se incluyen en el texto usando cite [5]. Para citar webs, artículos o libros [**?**].

[12]Créditos a los proyectos de Álvaro López Cantero: Configurador de Presupuestos y Roberto Izquierdo Amo: PLQuiz

3.3. Imágenes

Se pueden incluir imágenes con los comandos standard de LaTeX, pero esta plantilla dispone de comandos propios como por ejemplo el siguiente:

Figura 3.1: Autómata para una expresión vacía

3.4. Listas de items

Existen tres posibilidades:

- primer item.
- segundo item.

1. primer item.
2. segundo item.

Primer item más información sobre el primer item.

Segundo item más información sobre el segundo item.

-

3.5. Tablas

Igualmente se pueden usar los comandos específicos de LaTeXo bien usar alguno de los comandos de la plantilla.

Herramientas	App AngularJS	API REST	BD	Memoria
HTML5	X			
CSS3	X			
BOOTSTRAP	X			
JavaScript	X			
AngularJS	X			
Bower	X			
PHP		X		
Karma + Jasmine	X			
Slim framework		X		
Idiorm		X		
Composer		X		
JSON	X	X		
PhpStorm	X	X		
MySQL			X	
PhpMyAdmin			X	
Git + BitBucket	X	X	X	X
MikTEX				X
TEXMaker				X
Astah				X
Balsamiq Mockups	X			
VersionOne	X	X	X	X

Tabla 3.1: Herramientas y tecnologías utilizadas en cada parte del proyecto

Técnicas y herramientas

Esta parte de la memoria tiene como objetivo presentar las técnicas metodológicas y las herramientas de desarrollo que se han utilizado para llevar a cabo el proyecto. Si se han estudiado diferentes alternativas de metodologías, herramientas, bibliotecas se puede hacer un resumen de los aspectos más destacados de cada alternativa, incluyendo comparativas entre las distintas opciones y una justificación de las elecciones realizadas. No se pretende que este apartado se convierta en un capítulo de un libro dedicado a cada una de las alternativas, sino comentar los aspectos más destacados de cada opción, con un repaso somero a los fundamentos esenciales y referencias bibliográficas para que el lector pueda ampliar su conocimiento sobre el tema.

Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto, comentados por los autores del mismo. Debe incluir desde la exposición del ciclo de vida utilizado, hasta los detalles de mayor relevancia de las fases de análisis, diseño e implementación. Se busca que no sea una mera operación de copiar y pegar diagramas y extractos del código fuente, sino que realmente se justifiquen los caminos de solución que se han tomado, especialmente aquellos que no sean triviales. Puede ser el lugar más adecuado para documentar los aspectos más interesantes del diseño y de la implementación, con un mayor hincapié en aspectos tales como el tipo de arquitectura elegido, los índices de las tablas de la base de datos, normalización y desnormalización, distribución en ficheros3, reglas de negocio dentro de las bases de datos (EDVHV GH GDWRV DFWLYDV), aspectos de desarrollo relacionados con el WWW... Este apartado, debe convertirse en el resumen de la experiencia práctica del proyecto, y por sí mismo justifica que la memoria se convierta en un documento útil, fuente de referencia para los autores, los tutores y futuros alumnos.

Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

Bibliografía

[1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Patrones De Diseño: Elementos De Software Orientado a Objetos Reutilizable*. Addison-Wesley, 1 ed. en es edition.

[2] Ivar Jacobson, Grady Booch, and James Rumbaugh. *El proceso unificado de desarrollo de software*. Addison Wesley.

[3] Jacek Ratzinger. sPACE: Software project assessment in the course of evolution. http://www.inf.usi.ch/jazayeri/docs/Thesis_Jacek_Ratzinger.pdf.

[4] Ian Sommerville. *Ingeniería del software*. Pearson Education, 6ª edition.

[5] Wikipedia. Latex — wikipedia, la enciclopedia libre, 2015.

[6] Miguel Ángel León Bardavío. Evolution metrics gauge - comparador de métricas de evolución en repositorios software. https://gitlab.com/mlb0029/comparador-de-metricas-de-evolucion-en-repositorios-software.