UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Máster Universitario en Ingeniería en Informática

TFM del Máster Universitario en Ingeniería Informática

Análisis Visual de Revisiones de Código

Presentado por Mario Juez Gil
en Universidad de Burgos — 31 de marzo de 2017
Tutores: Carlos López Nozal, Raúl Marticorena Sánchez

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Máster Universitario en Ingeniería en Informática

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

Expone:

Que el alumno D. Mario Juez Gil, con DNI 71308224J, ha realizado el Trabajo final de Máster en Ingeniería Informática titulado Análisis Visual de Revisiones de Código.

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, 31 de marzo de 2017

Vº. Bº. del Tutor:

D. Carlos López Nozal

Vº. Bº. del co-tutor:

D. Raúl Marticorena Sánchez

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

Descripción del contenido del trabajo y del estrucutra de la memoria y del resto de materiales entregados.

Objetivos del proyecto

Este apartado explica de forma precisa y concisa cuales son los objetivos que se persiguen con la realización del proyecto. Se puede distinguir entre los objetivos marcados por los requisitos del software a construir y los objetivos de carácter técnico que plantea a la hora de llevar a la práctica el proyecto.

Conceptos teóricos

3.1. Conceptos relacionados con las revisiones de código

En este apartado se exponen una serie de conceptos teóricos relacionados con las revisiones de código. Se define cada concepto así como su relación con el presente trabajo.

Revisión de código

La revisión de código es un proceso de ingeniería a través del cual se realiza una inspección del código fuente por otros desarrolladores diferentes al autor del mismo. Resulta útil para reducir los defectos de código así como mejorar la calidad del software [1].

Frente a las revisiones de código altamente estructuradas propuestas por Fagan [5], hoy en día se están adoptando metodologías más livianas con el fin de solventar las ineficiencias de las inspecciones de código. Las denominadas *Modern Code Reviews* son informales, hacen uso de herramientas, y se utilizan regularmente.

Mediante el uso de estas nuevas metodologías, las revisiones de código ofrecen un mayor número de beneficios a los equipos de desarrollo como transferencia de conocimientos, visión de equipo, o mejores soluciones a los problemas [2].

Las revisiones de código son el elemento principal de este trabajo, donde se van a obtener datos de las mismas realizadas en diversos repositorios de software.

Inspección de software

La inspección de software, también conocida como inspección de Fagan, consiste en la búsqueda de defectos en documentos de desarrollo (código, especificaciones, diseño, etc.) por parte de personas con diversos roles mediante un proceso estructurado y bien definido, con el fin de disminuir el número de errores y aumentar la calidad del producto final [5].

Las fases de dicho proceso son las siguientes [6]:

Planificación En esta fase, los materiales que van a ser inspeccionados deben coincidir con los criterios de entrada de la inspección. Tiene lugar la elección de los participantes de la inspección. También se realiza la organización de las reuniones (hora y lugar).

Información general En esta fase se comunica a los participantes cuales son los resultados esperados. También se realiza la asignación de roles.

Preparación En esta fase tiene lugar el estudio del material por parte de los participantes y su preparación para cumplir sus roles.

Reunión de inspección Esta es la fase donde se lleva a cabo la búsqueda de defectos.

Repetición del trabajo El autor debe solucionar los defectos detectados, tras ello todo el proceso vuelve a comenzar desde la fase de planificación.

Seguimiento El moderador o equipo completo de inspección debe verificar que todos los cambios son correctos y no introducen nuevos defectos.

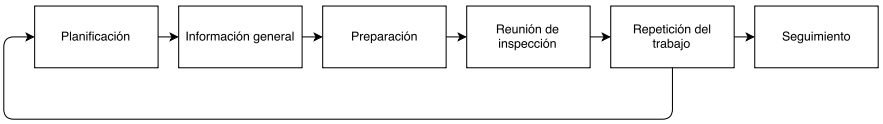

Figura 3.1: Operaciones de la inspección de software

Además, durante el proceso de inspección de software intervienen los siguientes roles:

Autor Responsable del desarrollo del documento a inspeccionar (código).

Lector Responsable de leer el documento como si fuese a desarrollarlo él mismo.

Revisor Responsable de examinar e inspeccionar el documento con el fin de identificar posibles defectos.

Moderador Responsable de coordinar la reunión de inspección.

Revisor de código

El revisor de código tiene como responsabilidad principal examinar el código y detectar posibles defectos, ofreciendo comentarios y sugerencias de cambio al autor del código que permitan la mejora de la calidad del mismo.

Los aportes del revisor de código también pueden proporcionar nuevos conocimientos al autor del código revisado, mejorando así sus habilidades lo cual se puede traducir en una mejora progresiva de la calidad del software que desarrolle.

Se pueden distinguir dos tipos de revisores de código:

Revisor experto Es una persona, idealmente con experiencia y amplios conocimientos (aunque no es estrictamente necesario).

Robot revisor Son herramientas que permiten la revisión automática de código mediante la comprobación del código fuente para garantizar que cumpla un conjunto de reglas predefinidas, así como detectar errores [10].

Actualmente es común que las revisiones cuenten con ambos tipos de revisores, en primer lugar se realiza una revisión automática, y los resultados son utilizados por el revisor experto para ofrecer una mejor revisión con un mayor nivel de detalle.

El revisor de código es una figura importante de este trabajo. Nos interesa extraer los comentarios que registra en cada cambio puesto que pueden resultar útiles para obtener métricas que podrían permitir evaluar la calidad de los comentarios o incluso del propio revisor.

Sistema de control de versiones

Un sistema de control de versiones se encarga de registrar los diferentes cambios de un fichero o un conjunto de ficheros a lo largo del tiempo, permitiendo así recuperar versiones específicas en cualquier momento [3].

Gracias a este tipo de sistemas es posible revertir los cambios realizados en un fichero o incluso en un proyecto completo a un estado anterior, comparar los cambios, comprobar la autoría de los mismos, etc.

Los sistemas de control de versiones están en constante evolución, actualmente herramientas como Github están empezando a implementar funcionalidades para facilitar las revisiones de código. Por tanto, en este trabajo vamos a utilizar este tipo de sistemas como fuente de datos a extraer.

Repositorio

El repositorio es la parte central de un sistema de control de versiones, es el lugar donde se almacenan los datos del sistema, normalmente estructurados en forma de árbol de ficheros [9].

La característica principal de un repositorio es que mantiene todas las versiones de cada fichero, permitiendo al cliente recuperar estados previos de los mismos.

Cambio (diff)

Un cambio es una modificación concreta de un fichero alojado en en un repositorio de un control de versiones [11].

Resulta útil para comparar dos versiones del mismo archivo.

Rama (branch)

Una rama es un apuntador a una confirmación. Por cada confirmación realizada, la rama avanza hasta la última confirmación. Por defecto existe una rama principal [3].

Crear otras ramas sirve para abrir nuevas líneas de desarrollo, por ejemplo una rama de pruebas, evitando así desordenar la principal.

Integración (merge)

Se entiende por integración a la unión de los cambios de dos ramas en una. Existen dos patrones de uso de esta operación [4]:

- Añadir los cambios realizados en una rama donde se están implementando nuevas características a la rama principal.
- Añadir los cambios de la rama principal en una rama donde se están implementando nuevas características con el fin de mantener la rama actualizada con los últimos parches y características. Esta práctica permite reducir el riesgo de conflictos[1] al unir la rama de desarrollo con la principal.

Confirmación (commit)

Una confirmación (o commit) supone el almacenamiento en el repositorio de una nueva versión de los ficheros que contiene, en otras palabras, sirve para guardar los cambios [3].

[1] Existe un conflicto cuando al integrar dos ramas, un mismo fichero ha sido modificado en ambas y por ello es necesario decidir que cambios deben mantenerse en la versión final.

Cada confirmación lleva asociados uno o varios ficheros modificados, un identificador denominado revisión y un mensaje donde se describen los cambios realizados.

Etiqueta (tag)

La finalidad del etiquetado es dar nombre a alguna versión para que pueda ser localizada facilmente en el futuro, es decir, permite identificar de forma sencilla revisiones importantes del proyecto [11].

Pull request

Las "Pull Requests"son una característica concreta de la herramienta Github. Sirven para mantener conversaciones sobre los cambios donde se pueden exponer ideas, asignar tareas, tratar detalles y hacer revisiones de código [8].

3.2. Conceptos relacionados con buenas prácticas ágiles

Integración continua

La integración continua es una práctica de desarrollo software donde los miembros de un equipo integran su trabajo frecuentemente, normalmente como mínimo una integración diaria [7].

Cada integración se verifica mediante la automatización de ciertos procesos como construcción, pruebas y despliegue. Mediante este tipo de verificación se busca detectar errores de integración tan rápido como sea posible.

Para el desarrollo de este trabajo se utilizarán técnicas de integración continua.

Despliegue continuo

Construcción automática

Diagrama "Burn Down"

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] A. F. Ackerman, L. S. Buchwald, and F. H. Lewski. Software inspections: an effective verification process. *IEEE Software*, 6(3):31–36, May 1989.

[2] Alberto Bacchelli and Christian Bird. Expectations, outcomes, and challenges of modern code review. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 712–721, Piscataway, NJ, USA, 2013. IEEE Press.

[3] Scott Chacon and Ben Straub. *Pro Git*. Apress, Berkely, CA, USA, 2nd edition, 2014.

[4] Charles Duan. Understanding Git: merging. https://www.sbf5.com/~cduan/technical/git/git-3.shtml. [Internet; accedido 14-marzo-2017].

[5] M. E. Fagan. Design and code inspections to reduce errors in program development. *IBM Systems Journal*, 15(3):182–211, 1976.

[6] Michael E. Fagan. Advances in software inspections. *IEEE Trans. Softw. Eng.*, 12(1):744–751, January 1986.

[7] Martin Fowler and Matthew Foemmel. Continuous integration. *Thought-Works) http://www. thoughtworks. com/Continuous Integration. pdf*, page 122, 2006.

[8] Github. Github features. https://github.com/features. [Internet; accedido 15-marzo-2017].

[9] C Pilato, Ben Collins-Sussman, and Brian Fitzpatrick. *Version Control with Subversion*. O'Reilly Media, Inc., 2 edition, 2008.

[10] Wikipedia. Revisión automática de código — wikipedia, la enciclopedia libre, 2014. [Internet; accedido 9-marzo-2017].

[11] Wikipedia. Control de versiones — wikipedia, la enciclopedia libre, 2017. [Internet; accedido 10-marzo-2017].