Tribalyte | Álvaro Gómez Haro | Julio 2020

La importancia de la revisión de código como herramienta de aprendizaje y mejora de la calidad en el desarrollo de software.

Un análisis de la literatura sobre las ventajas de la revisión cruzada de código fuente.

Índice

Resumen

En este documento se mostrará un análisis de la literatura acerca de las ventajas de la revisión de código para la mejora de calidad, reducción de costes y riesgos, y la eliminación de parte de la volatilidad de los equipos humanos de desarrollo software. También, se realizará un pequeña enumeración de los estándares que contienen la *"revisión por pares"* o *"revisión cruzada"* como una práctica *cuasi* obligatoria en el desarrollo software.

Introducción

La revisión de código es una herramienta conocida por gran parte de la industria del desarrollo de software. Sin embargo, como se menciona en el texto de Karl E. Wiegers [1]:

"... muchas organizaciones tienen dificultades implementado un programa de revisiones efectivo. Gran parte de las barreras contra la revisión por pares son de naturaleza social y cultural, y no técnica."

Para poder superar estas barreras se deben conocer las ventajas de la revisión por pares, no solo como una herramienta para la organización que la desee implantar, sino como una actividad formativa de mejora contínua tanto para los revisores como para los autores del código en cuestión. Las revisiones recibidas no deben de ser interpretadas como defectos o fallos en el código del autor, sino como posibles mejoras o sugerencias que ofrece el revisor. De igual manera el revisor no debe apreciarlas como un trabajo que hace por el autor, sino tiempo que ahorra al equipo en el futuro, por ejemplo identificando y corrigiendo bugs o evitando rehacer funcionalidades [2].

Con el objetivo de derribar estas barreras, a continuación, se presenta un análisis en mayor profundidad de las ventajas de este proceso.

Ventajas de la revisión

Mayor atención a las buenas prácticas

Debido al conocimiento de que su código será revisado, los desarrolladores se ven motivados a resolver de la manera más adecuada, todas y cada una de las características de su implementación, siendo esto fuente de orgullo en el gremio, y así, mejorando el buen hacer de los individuos [3].

Fomento del trabajo en equipo

Las revisiones a veces plantean un conflicto entre el autor y el revisor [4], este conflicto debe ser tratado como sugerencia y no como ataque, sin embargo, esta disputa de intereses favorece el entendimiento de los integrantes del equipo y les ayuda a resolver de manera efectiva, rápida y pacífica todo tipo de retos y desacuerdos.

Transferencia de conocimiento

Cuando los miembros de un equipo de desarrollo leen y comentan el código de sus compañeros, adquieren un mayor conocimiento del proyecto y de las tecnologías en uso, mejorando así su preparación a la hora de asumir las siguientes tareas dentro del mismo proyecto o en otros proyectos similares..

Altamente efectivo para evitar "bugs"

Según Steve McConnell, en su conocida obra *"Code Complete: A Practical Handbook of Software Construction"* [5]:

"...la media de detección de errores es de tan solo un 25% para los test unitarios, 35% para los tests funcionales y

45% para los tests de integración. Sin embargo, la revisión de diseño y código se encuentra entre un 55% y un 60% de efectividad media."

Ahorro de tiempo y dinero

Un código de calidad, legible y fácilmente mantenible reduce el tiempo y coste de desarrollos futuros. Mediante la revisión de código se establecen unas guías de estilo y buenas prácticas comunes de facto, lo cual minimiza el impacto de la rotación del talento en el equipo, evitando como consecuencia la creación de desarrolladores críticos [2] en ciertas áreas o funcionalidades, y otorgando así mayor libertad a estos últimos para solicitar vacaciones o trabajar en diferentes tareas.

Mejora de la seguridad

Con la revisión por pares, en cada cambio o "commit" se aplica el conocimiento del autor y todos los revisores. Si al menos uno de los participantes, tiene conocimientos de seguridad, o ha oído hablar de algún *"exploit"* que puede afectar al código actual, el código quedará modificado para evitar estos problemas y todos los participantes mantendrán parte de ese conocimiento para próximos problemas similares [6], mejorando así la seguridad del proyecto de manera "pasiva".

Estándares de calidad

Añadido a todas estas ventajas, la revisión por pares forma parte de los procesos necesarios para la certificación en múltiples estándares de aseguramiento de calidad en el desarrollo de software.

Los ligados directamente con la revisión por pares y calidad del código son la ISO/IEC 20246 [7] en su completitud y la CMMI-DEV [8] en el apartado de *"Verificación"*. Para continuar con la ISO/IEC 25041 [9], mencionando la revisión de código como herramienta de evaluación, la ISO/IEC 14598 [10] cubriendo la revisión de código para desarrolladores y, por último, utilizada para aplicar y certificar con la ISO/IEC 9126 [11].

Conclusión

La revisión de código es una de las herramientas más útiles y efectivas para lograr un equipo de desarrollo concienciado por la calidad de código, mejores prácticas de programación y estándares de calidad, que además son certificables. Añadir la revisión cruzada a los procesos internos de la organización contribuye a crear productos más fiables, reduciendo los riesgos durante sus desarrollos. Además, no solo beneficia en el ámbito de la organización o el del equipo, sino que también como entrenamiento de cada uno de los participantes en las revisiones, en su proceso individual de mejora continua.

Múltiples autores [1,3,5,7] proponen establecer guías de revisión y estilo, de manera que los participantes conozcan lo que buscar, o lo que evitar, en sus procesos de revisión y autoría. Además, aportar un conjunto de procedimientos y consejos para la revisión y/o la autoría puede facilitar la integración del proceso en el flujo de trabajo.

Lograr implementar una cultura de revisión y percepción de la calidad del código dentro de la organización puede a ser un gran reto [1] muchas veces debido a la falta de comprensión de los beneficios de este proceso por parte de los desarrolladores.

Por ello, existen múltiples herramientas que facilitan la revisión de código. Entre ellas, sistemas de control de versiones como Github [12], Bitbucket [13] o Gitlab [14], las más conocidas, junto con otras como Gerrit [15] o CodeStriker [16]. Además, existen herramientas de gestión de la reputación y gamificación del desarrollo de software, como por ejemplo *BrightByte* [17], la cual ameniza el proceso de revisión, convirtiéndolo en un foco de entretenimiento y orgullo para los desarrolladores, y con ello, ayudando a incorporar la conciencia de la calidad del código a la cultura de la organización.

Bibliografía

[1] Karl E. Wiegers, *"Humanizing Peer Reviews"*, https://www.processimpact.com/articles/humanizing_reviews.pdf

[2] Dan Radigan, *"Why code reviews matter (and actually save time!)"*, https://www.atlassian.com/agile/software-development/code-reviews

[3] Robert F, *"Code Review Best Practices"*, https://medium.com/palantir/code-review-best-practices-19e02780015f

[4] Derek Prior, *"Implementing a Strong Code-Review Culture"*, RailsConf 2015 https://www.youtube.com/watch?v=PJjmw9TRB7s

[5] Steve McConnell, *"Code Complete: A Practical Handbook of Software Construction"*

[6] Bruce Johnson, *"Lessons From Google: How Code Reviews Build Company Culture"*, https://blog.fullstory.com/what-we-learned-from-google-code-reviews-arent-just-for-catching-bugs/

[7] ISO/IEC 20246, *"Software and systems engineering - Work product reviews"*, https://www.sis.se/api/document/preview/921568/

[8] CMMI-DEV, *"Mejora de los procesos para el desarrollo de mejores productos y servicios"*, https://cmmiinstitute.com/getattachment/4439387f-28aa-4f3a-8f2b-a0cc5b449e47/attachment.aspx

[9] ISO/IEC 25041, *"Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE)"*, https://www.iso.org/obp/ui/#iso:std:iso-iec:25041:ed-1:v1:en

[10] ISO/IEC 14598, *"Tecnología de la información. Evaluación del producto software"*, https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma?c=N0034121

[11] ISO/IEC 9126, *"Ingeniería del software. Calidad del producto software"*, https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma/?c=N0032555

[12] Github, https://github.com/

[13] Bitbucket, https://bitbucket.org/

[14] Gitlab, https://about.gitlab.com/

[15] Gerrit, https://www.gerritcodereview.com/

[16] CodeStriker, http://codestriker.sourceforge.net/

[17] BrightByte, http://brightbyteapp.com/

Otras referencias

- Jeff Atwood, *"Code Reviews: Just Do It"*, https://blog.codinghorror.com/code-reviews-just-do-it/

- Smart Bear, *"10 tips to guide you toward effective peer code review"*, https://smartbear.com/learn/code-review/best-practices-for-peer-code-review/

- Madalin Ilie, *"Code review guidelines"*, *https://www.codeproject.com/Articles/524235/Codeplusreviewplusguidelines*

- Google-eng, *"Google's Engineering Practices documentation"*, https://google.github.io/eng-practices/review/reviewer/

- Perforce, *"Coding Standards For Quality and Compliance"*, https://www.perforce.com/resources/qac/coding-standards