Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 350-365

Tipo de artículo: Artículo original
Temática: Ingeniería y gestión de software
Recibido: 30/06/2021| Aceptado: 01/10/2021

Aplicación de un proceso para la gestión de la mantenibilidad en el desarrollo de software

Application of a process for the management of maintainability in software development

Lisandra Tamayo Espinosa [1*] https://orcid.org/0000-0003-1067-0983
Nemury Silega Martínez[1] https://orcid.org/0000-0002-8436-5650

[1]Universidad de las Ciencias Informáticas. Carretera a San Antonio de los Baños, Km 2 ½, Torrens, Boyeros, La Habana, Cuba. E-mail: {ltespinosa, nsilega}@uci.cu

*Autor para la correspondencia. (ltespinosa@uci.cu)

RESUMEN

La inadecuada atención a la mantenibilidad desde etapas tempranas provocará que la mayor parte del tiempo que dedican los programadores al desarrollo, sea invertido en la etapa de mantenimiento. Además, para obtener un producto de software de calidad, es indispensable la ejecución de acciones que garanticen una correcta gestión de la mantenibilidad y disminuyan los altos costos del mantenimiento de los sistemas una vez sean entregados al cliente. A partir de la importancia de esta característica para el éxito en la mejora de los procesos de software, se elaboró un proceso con las actividades a realizar en las disciplinas ingenieriles de desarrollo, así como los roles y artefactos de entrada y salida a estas. Luego, se aplicó el método estudio de casos en un proyecto de desarrollo de software, con el objetivo de obtener las diferencias

Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 350-365

a partir de la comparación de métricas de mantenibilidad, antes y después de ejecutado el proceso. Los resultados obtenidos fueron satisfactorios, ya que tanto la complejidad como la densidad de código duplicado disminuyeron luego de aplicado el proceso.

Palabras clave: mantenibilidad; proceso; métricas; complejidad; código duplicado.

ABSTRACT

The inadequate attention to maintainability from early stages will cause that most of the time that programmers dedicate to development will be invested in the maintenance stage. Furthermore, in order to obtain a quality software product, it is essential to implement actions that ensure proper maintainability management and reduce the high costs of maintaining the systems once they are delivered to the customer. Based on the importance of this characteristic for the success in the improvement of software processes, a process was elaborated with the activities to be performed in the development engineering disciplines, as well as the roles and input and output artifacts. Then, the case study method was applied in a software development project, with the objective of obtaining the differences from the comparison of maintainability metrics, before and after the process was executed. The results obtained were satisfactory, since both the complexity and the density of duplicated code decreased after the process was applied.

Keywords: maintainability; processes; metrics; complexity; duplicate code.

Introducción

La mantenibilidad tiene un impacto significativo en la calidad general de una aplicación (Rodríguez, Pedreira, & Fernández, 2015). Puede ser uno de los atributos de calidad más difíciles de estimar, porque esto implica inherentemente hacer predicciones sobre actividades futuras, por lo que podemos afirmar que

es difícil de medir ya sea de manera cualitativa o cuantitativa (Döhmen, Bruntink, Ceolin, & Visser, 2016). El Glosario del Estándar IEEE (IEEE, 1990) define la mantenibilidad como: la facilidad con la que un sistema o componente de software puede modificarse para corregir fallos, mejorar el rendimiento u otros atributos, o adaptarse a un entorno cambiado.

La correcta gestión de la mantenibilidad incrementa las facilidades de modificación, al obtener productos de software con documentación actualizada y suficientes comentarios en el código. Provee beneficios en la reducción de los costos y recursos destinados a la fase de mantenimiento, gracias a la disminución en la complejidad y la densidad de código repetido. Tratar la mantenibilidad desde las fases tempranas del ciclo de vida aumenta el valor de los proyectos, debido a su influencia sobre los recursos de desarrollo y sobre la etapa futura de mantenimiento. Permite obtener un software fácilmente mantenible y con un código de alta calidad. Además, propicia la adaptación del producto a los cambios propios de la evolución en el desarrollo de software (Erazo, Florez, & Pino, 2016).

El aumento de la mantenibilidad es un requisito económico importante para las empresas con grandes sistemas de información de larga duración (English, Buckley, & Collins, 2016). Con esta premisa se desarrolló un proceso para la gestión de la mantenibilidad desde etapas tempranas en el desarrollo de software. Logrando con su aplicación un impacto positivo en la disminución de los costos en los cambios que se producen a los requisitos, el diseño, la arquitectura y el código. Reconociendo, además, que aumenta las capacidades de reutilización, obteniendo códigos con baja complejidad y disminuyendo los duplicados. Todo ello, logrado gracias a la inclusión de estos dos factores de mantenibilidad desde las fases tempranas del desarrollo.

Aquellos factores que pueden medirse son reconocidos como atributos o métricas de mantenibilidad. Los estudios realizados por Morasca exponen que las métricas son un medio para entender, monitorizar, controlar, predecir y probar el desarrollo del software (Morasca, 1996). Una métrica es una característica de un producto, documentación de sistema o proceso de desarrollo que puede medirse de manera objetiva. También las métricas son usadas para evaluar los atributos internos de un producto de software (Mesias, 2018).

En el presente artículo se analizan las métricas complejidad y código duplicado, así como la herramienta SonarQube usada para evaluar estas métricas de mantenibilidad. Los resultados son un fragmento del

diagnóstico utilizado en la confección de un proceso para la gestión de la mantenibilidad desde etapas tempranas en el desarrollo de software. Como parte de un estudio de casos, se decidió aplicar el proceso desarrollado en uno de los proyectos evaluados inicialmente. Luego se realiza una segunda iteración de pruebas para analizar el impacto del proceso propuesto. Teniendo en cuenta el mismo periodo de tiempo, se evalúa también el comportamiento de la mantenibilidad en un segundo caso de estudio.

Métodos o Metodología Computacional

Se identificaron 30 publicaciones cuyos temas especificaban la mantenibilidad, de ellas se consideraron relevantes para la presente investigación 18. De la revisión bibliográfica resaltaron cinco artículos de los autores Irrazábal, Erazo y Rodríguez, que presentaban diferentes propuestas para evaluar, certificar y gestionar la mantenibilidad. En estas publicaciones se examinaban las sub-características de mantenibilidad planteadas por la ISO/IEC 25010: 2011, los factores y buenas prácticas para el proceso de desarrollo de software, teniendo en cuenta el aumento de la mantenibilidad. Estos elementos sirvieron de base para la elaboración del proceso para la gestión de la mantenibilidad que se aplica en el estudio de casos. El uso de la herramienta SonarQube para el diagnóstico de las métricas complejidad y densidad de duplicados, antes y después de aplicado el proceso, permitió una valoración del impacto positivo del proceso elaborado.

Resultados y discusión

Es fundamental conocer la situación en la que se encuentra el producto de software lo más rápido posible para poder decidir mejor cómo gestionar y realizar cada petición de mantenimiento, y en general planificar mejor el proceso (Ruiz & Polo, 2019). La medición de software es el proceso mediante el cual se puede obtener un valor numérico a partir de un proceso o producto de software. Las del producto son mediciones que se realizan en cualquier estado de su desarrollo, desde los requerimientos hasta el sistema instalado y son ampliamente usadas por la industria del software (Pérez, 2015).

Para Rodríguez (Rodríguez et al., 2015), no solo es importante definir las sub-características de mantenibilidad, sino que el desglose de estas al más bajo nivel, donde se definen métricas que se pueden medir a través de los productos de software. A pesar de que la mantenibilidad del software es una tarea costosa y desafiante, a menudo está mal gestionada. Una de las razones de la mala gestión es la falta de medidas probadas (Aggrawal, 2002). A continuación, se analizan las métricas que hacen un aporte significativo a la investigación:

1. Complejidad: Se relaciona con la dificultad para implementar, probar, entender, modificar o mantener un programa. Está relacionada con la analizabilidad, modificabilidad y testabilidad. En aplicaciones con complejidad elevada, las tareas de mantenimiento requieren mayor esfuerzo y, por tanto, son más costosas (Rodríguez & Piattini, 2014).
2. Código duplicado: Hace referencia a fragmentos de código que se encuentran repetidos en diferentes lugares del sistema. El código duplicado hace más difícil la realización de modificaciones en una aplicación, ya que solucionar un defecto o introducir una mejora en el código, requiere realizar las modificaciones en todas las partes del sistema en las que aparezca. Afecta a las sub-características de mantenibilidad: analizabilidad y modificabilidad, ya que obstaculiza la identificación del código que se debe modificar y la realización de modificaciones que sean efectivas. También dificulta la testabilidad, pues se deben definir varios casos de prueba diferentes, para comprobar el correcto funcionamiento del código duplicado en sus distintas instancias.

Una métrica básica de calidad es el código repetido, que junto a la complejidad constituyen el mayor enemigo de la mantenibilidad (Valenciano, 2015; Rojas & Alvarez, 2020). A su vez un software con un nivel adecuado de mantenibilidad aporta entre otros beneficios la simplificación de la complejidad (AQCLab, 2019). Teniendo estas premisas en mente, y el hecho de que en la actualidad los desarrollos de software suelen tener un gran tamaño, por lo que el número de medidas a obtener es muy elevado, parece que el enfoque más conveniente sería disponer de herramientas que facilitaran la obtención automática de tales medidas.

Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 350-365

SonarQube para medir la mantenibilidad

SonarQube se utiliza para medir la mantenibilidad del código fuente y gestionar su deuda técnica. Puede detectar fugas de memoria, casos avanzados de derivaciones de punteros nulos, condiciones que son siempre verdaderas o falsas, independientemente de la ruta de ejecución (Sonarqube, 2019). los resultados muestran que al incorporar sonarqube al desarrollo se mejora la mantenibilidad (Carvajal, Tasis, & Martínez, 2019). se utiliza para evaluar código fuente, brinda gran cantidad de funcionalidades para la detección de código duplicado, no adecuación a estándares y convenciones de código, vulnerabilidades conocidas de seguridad, complejidad ciclomática y acoplamiento (León, 2020).

SonarQube es una plataforma de código abierto, usada por los equipos de desarrollo para controlar la calidad del código. Fue desarrollada con el principal objetivo de hacer accesible la administración de la calidad del código con un mínimo esfuerzo. Como tal, SonarQube contiene en su núcleo de funcionalidades un analizador de código, una herramienta de reportes, un módulo que detecta defectos y una función para regresar los cambios realizados en el código. Presenta los valores en una interfaz agradable y fácil de comprender, por ejemplo: para representar el código duplicado establece colores según el rango de valores, aquellos programas que cuentan con un valor por debajo del 5% los representa en color verde, entre 5% y 20% en color naranja y los que están por encima del 20% los señala en color rojo.

Descripción del proceso para la gestión de la mantenibilidad

Se optó por la creación de un proceso con el objetivo de gestionar la mantenibilidad desde etapas tempranas en el desarrollo de software, luego del análisis de los resultados obtenidos con la aplicación de los métodos científicos. El proceso incluye los factores que influyen en las sub-características planteadas por la Norma Cubana ISO/IEC 25010: 2016. El proceso describe las actividades en las fases: Inicio, Ejecución y Cierre; y las disciplinas ingenieriles propuestas por la metodología AUP-UCI (Informáticas, 2015): Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas, Pruebas de liberación y Pruebas de aceptación. Además, se detallan: roles, artefactos de entrada, actividades y artefactos de salida, basados en modelos y procesos acorde al ciclo de vida de un proyecto de desarrollo de software (Espinosa, 2021). El proceso propone actividades para asegurar la gestión de los requisitos no funcionales de mantenibilidad

Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 350-365

(RNFM). En las Figuras 1 y 2 se presentan las actividades correspondientes a la Fase de Inicio y a las disciplinas: Modelado de negocio y Requisitos, pertenecientes a la fase de Ejecución.

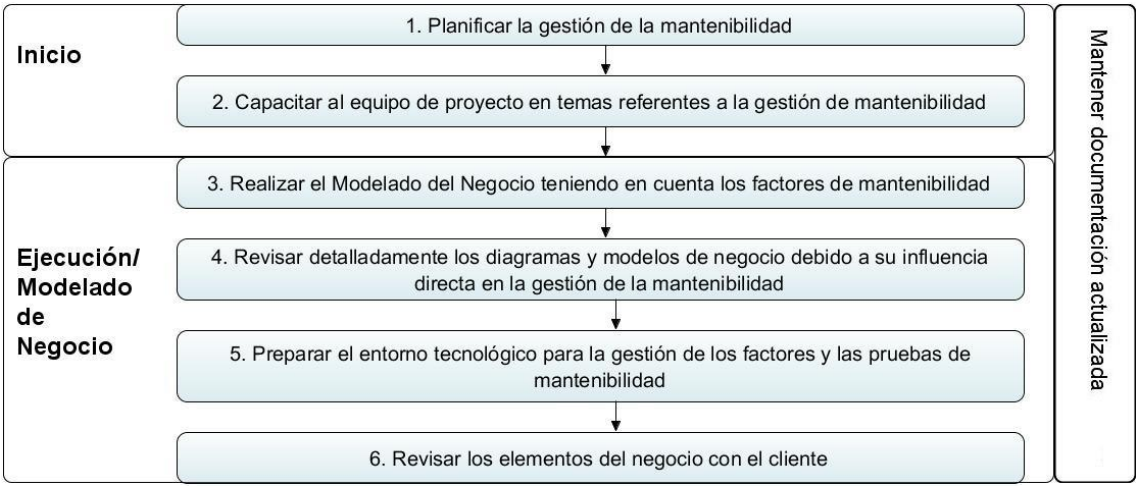

Fig. 1 – Actividades de Inicio y Modelado de negocio.

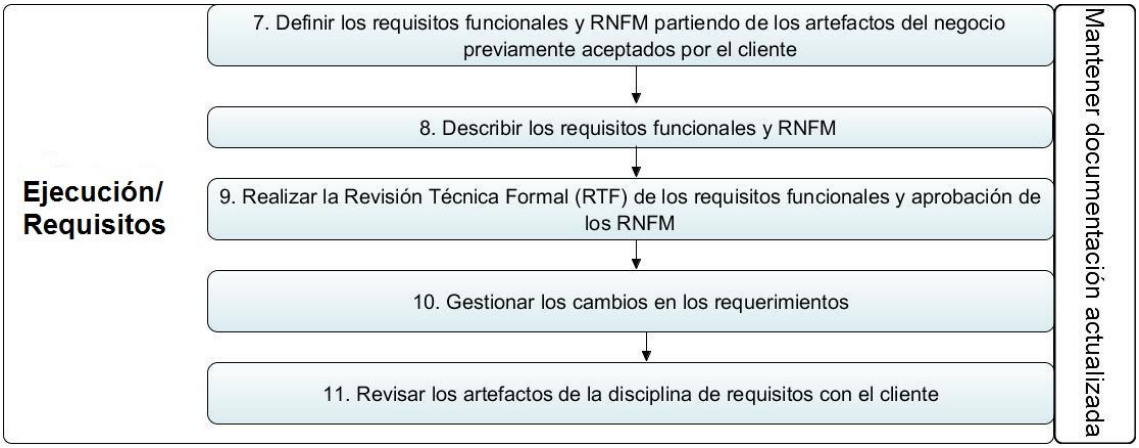

Fig. 2 – Actividades de las disciplinas Análisis y diseño e Implementación.

En la Figura 3 se presentan las actividades correspondientes a las disciplinas: Análisis y diseño e Implementación, pertenecientes a la fase de Ejecución.

Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 350-365

Fig. 3 – Actividades de las disciplinas Pruebas internas y Pruebas de liberación.

En las Figuras 4 y 5 se observan las actividades de las disciplinas: Pruebas internas, Pruebas de liberación y Pruebas de aceptación, así como la actividad de la fase de Cierre.

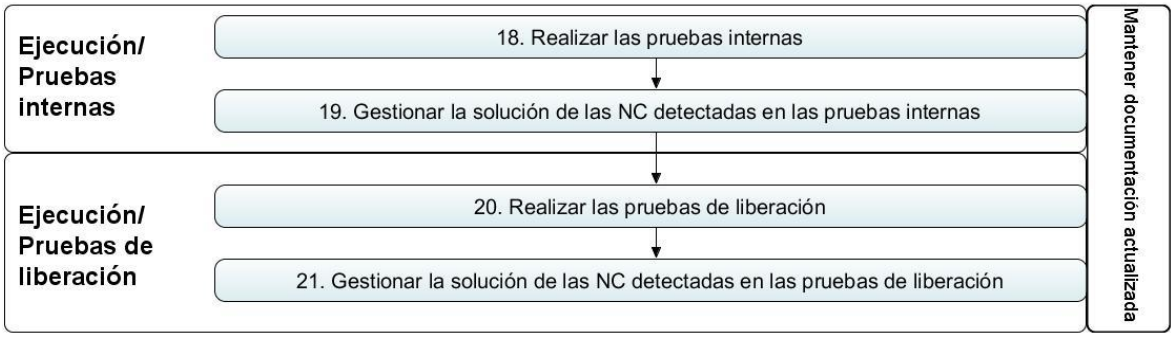

Fig. 4 – Actividades de las disciplinas Pruebas internas y Pruebas de liberación.

Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 350-365

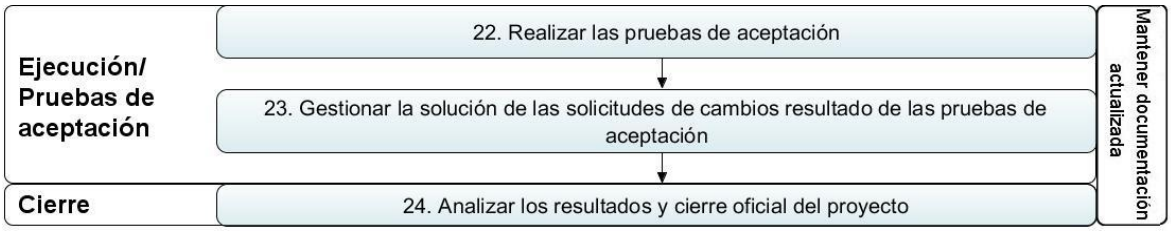

Fig. 5 – Actividades de la disciplina Pruebas de aceptación y de la fase de Cierre.

En la Tabla 1 se describen las actividades correspondientes a la fase de Inicio del proceso elaborado. Además, se definen los roles, artefactos de entrada y salida a estas actividades.

Tabla 1 - Descripción de la fase Inicio del proceso para la gestión de la mantenibilidad.

Fase /Disciplina	**Rol**	**Entrada**	**Actividad**	**Salida**
Inicio	Jefe de proyecto Administrador de la calidad	- Acta de inicio	1. Planificar la gestión de la mantenibilidad. Definir un plan de mantenibilidad donde se creen las actividades y tareas concernientes a su gestión.	- Plan de mantenibilidad - Cronograma del proyecto
	Jefe de proyecto Equipo de desarrollo Arquitecto Administradores Analista CCC	- Plan de mantenibilidad - Cronograma del proyecto	2. Capacitar respecto a los temas de mantenibilidad necesarios para su gestión. Presentar el plan de mantenibilidad. Explicar las responsabilidades y tareas de cada rol. Exponer los factores que influyen en las sub-características de mantenibilidad y su clasificación por disciplinas ingenieriles.	- Acta de compromiso con el Cronograma del proyecto - Plan de mantenibilidad

Aplicación del método estudio de casos

Con una investigación de estudio de casos se pueden lograr diferentes objetivos: hacer una descripción, ofrecer explicaciones o interpretaciones sobre el fenómeno investigado, explorar sus características, funcionamiento o hacer una evaluación (Monge, 2010). Se decidió aplicar este método al proyecto P1 para obtener diferencias respecto al comportamiento de las métricas complejidad y código duplicado. Se realizó

Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 350-365

la primera iteración de prueba para el estudio de la problemática con la herramienta SonarQube, luego se aplicó el proceso para la gestión de la mantenibilidad y se efectuó la segunda iteración de pruebas. Los valores obtenidos en las pruebas realizadas al primer caso P1 se comparan con los resultados del análisis realizado al segundo caso nombrado en la investigación como P2.

Los resultados del primer diagnóstico (ver Figura 6) arrojaron que la densidad de duplicados (relación entre la cantidad de código duplicado de un producto y su tamaño) era de 3.0. Mientras que la complejidad era de 16,229.

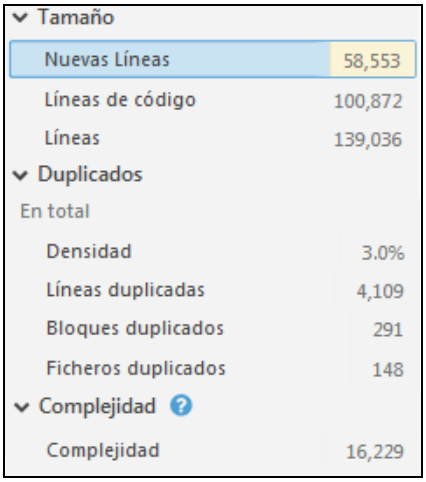

Fig. 6 – Valores del proyecto P1 antes de ejecutado el proceso.

En la versión Acompañamiento y Desarrollo de solicitudes de cambios de P1, se empleó el proceso para la gestión de la mantenibilidad. Para valorar el comportamiento de las métricas después de aplicado el proceso, se hicieron nuevamente pruebas con la herramienta SonarQube, obteniéndose una densidad de duplicados de 2.4 y una complejidad de 16,111 (ver Figura 7).

Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 350-365

Fig. 7 – Valores del proyecto P1 después de aplicado el proceso.

Los resultados obtenidos después de ejecutado el proceso fueron favorables, ya que tanto la complejidad como la densidad de código duplicado disminuyeron. Reconociendo, además, que estas métricas constituyen los peores enemigos de la mantenibilidad, se puede concluir que su disminución influye positivamente en el aumento de la mantenibilidad del producto de software obtenido.

Para evaluar el comportamiento de ambas métricas en otros espacios, se decidió valorar también con la herramienta SonarQube al proyecto P2. Este caso fue escogido debido a que en la primera iteración de pruebas, presentaba el menor valor en complejidad de los proyectos estudiados. En aquel momento contaba con una densidad de duplicados de 4.9 y una complejidad de 3, 005.

Luego en la segunda iteración de pruebas con la misma herramienta, para el mismo periodo de tiempo que el proyecto P1 y sin aplicar el proceso, P2 mostró valores desfavorables. La densidad de duplicados aumentó a 10.8 y la complejidad a 4, 128 (ver Figura 8). Si comparamos los valores recogidos en ambas iteraciones de pruebas, se puede concluir que en el proyecto P2 las métricas interpretadas ascendieron significativamente.

Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 350-365

Fig. 8 – Valores del proyecto P2 en la segunda prueba.

Después de explorar el comportamiento de las métricas en los proyectos P1 y P2, para el mismo periodo de análisis y utilizando la misma herramienta de prueba, se confirma el éxito del proceso elaborado. Como se reconoció durante la revisión bibliográfica, a medida que el programa de software evoluciona se vuelve más complejo y el código duplicado aumenta, si no se gestiona correctamente la mantenibilidad. Mediante el método estudio de casos esta premisa se ratificó (ver Figura 9 y Figura 10), pues las métricas código duplicado y complejidad disminuyeron en el proyecto donde se aplicó el proceso elaborado. Mientras, en el programa P2, donde no se utilizó el proceso para la gestión de la mantenibilidad, estas métricas aumentaron significativamente.

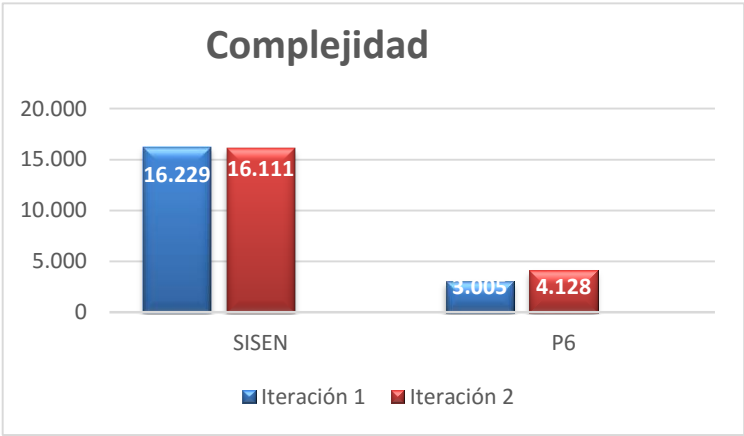

Fig. 9 – Comportamiento de la complejidad en los proyectos P1 y P2.

Revista Cubana de Ciencias Informáticas
Vol. 15, No. Especial UCIENCIA I, Septiembre, 2021
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 350-365

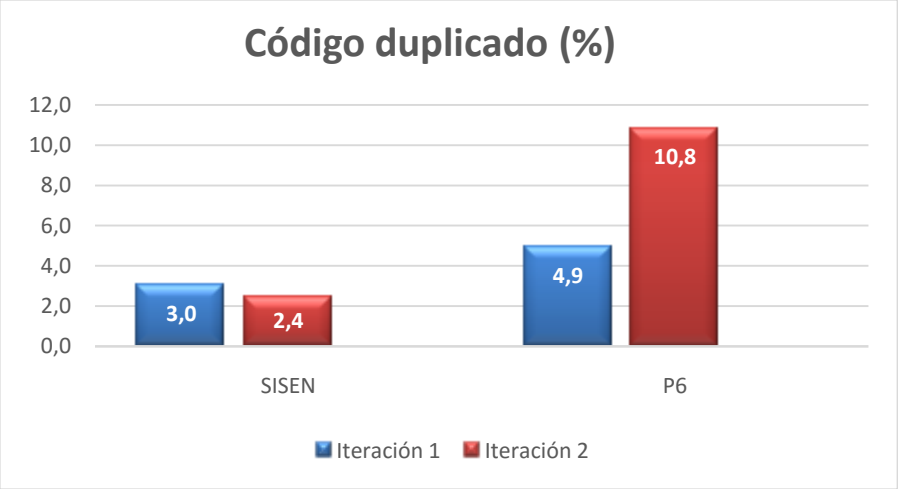

Fig. 10 – Comportamiento del código duplicado en los proyectos P1 y P2.

Conclusiones

Mediante el método estudio de casos se pudo conocer el estado de la mantenibilidad y el impacto positivo del uso del proceso elaborado para la gestión de la misma. Gracias a la evaluación realizada usando la herramienta SonarQube se observó que:

1. A medida que avanza un proyecto la complejidad aumenta si no se gestiona correctamente.
2. Junto con la complejidad aumenta también la dificultad de probar, lo que se tendría presente a la hora de planificar pruebas futuras.
3. Si no se tiene en cuenta la mantenibilidad desde etapas tempranas del desarrollo, entonces el producto resultante es difícil de mantener.
4. La mantenibilidad es una tarea costosa, sobre todo si no aplicamos métodos para su gestión.
5. Las métricas complejidad y código duplicado permiten hacer un estudio del software a medida que evoluciona.

Referencias

Aqclab. Evaluación Y Certificación De La Mantenibilidad Iso/Iec_25000. [En Línea], 2019. [Consultado El 20/02/2019]. Disponible En: [Http://Www.Aqclab.Es/Index.Php/Evaluacion-Certificacion-Calidad-Software-Iso-25000/Evaluacion-Certificacion-Mantenibilidad-Iso-25000].

Castro Monge, E. El Estudio De Casos Como Metodología De Investigación Y Su Importancia En La Dirección Y Administración De Empresas. Revista Nacional De Administración, 2010.

Döhmen, T., Bruntink, M., Ceolin, D., & Visser, J. Towards A Benchmark For The Maintainability Evolution Of Industrial Software Systems. Conference Of The International Workshop, 2016.

English, M., Buckley, J., & Collins, J. J. Investigating Software Modularity Using Class And Module Level Metrics, 2016.

Erazo, J., Florez, A., & Pino, F. Análisis Y Clasificación De Atributos De Mantenibilidad Del Software: Una Revisión Comparativa Desde El Estado Del Arte. Entre Ciencia E Ingeniería, 2016, 10(19): 40-49.

Espinosa, L. T. Gestión De La Mantenibilidad Desde Etapas Tempranas En El Desarrollo De Software. Revista Cubana De Ciencias Informáticas, 2021.

Gonzalez, Y. C., Gonzalo, M., & Manso, E. Mantenibilidad Y Productividad En La Enseñanza De La Ingeniería Del Software: Análisis Cuantitativo De Un Enfoque Práctico. Departamento De Informática, Universidad De Valladolid, España, 2019.

Ieee. Ieee Standard Glossary Of Software Engineering Terminology. Institute Of Electrical And Electronics Engineers, 1990.

Informáticas, U. D. L. C. Metodología De Desarrollo Para La Actividad Productiva Uci. 2015.

K. K Aggrawal, Y. S. A. J. K. C. An Integrated Measure Of Software Maintainability. In Proceedings Of Reliability And Maintainability Symposium, 2002.

Pardo Mesias, S. R. Mantenibilidad De Productos De Software Según El Modelo Iso/Iec 25000. Universidad Nacional Agraria De La Selva, 2018.

Pérez, H. G. Analizando La Mantenibilidad Del Software En Un Estudio Dentro La Formación De Estudiantes Universitarios. Revista Latinoamericana De Ingeniería De Software, 2015.

Rodríguez, M., Pedreira, O., & Fernández, C. M. Certificación De La Mantenibilidad Del Producto Software: Un Caso Práctico. Revista Latinoamericana De Ingeniería De Software, 2015, 3(3): P. 127-134.

Rodríguez, M., & Piattini, M. Entorno Para La Evaluación Y Certificación De La Calidad Del Producto Software. Jornadas Sistedes 2014 Cádiz, Del 16 Al 19 De Septiembre.Xix Jornadas De Ingeniería Del Software Y Bases De Datos, 2014.

Ruiz, F., & Polo, M. Mantenimiento Del Software. [En Línea], 2019. [Consultado El 20/02/2019]. Disponible En: Https://Alarcos.Esi.Uclm.Es/Per/Fruiz/Curs/Mso/Trans/S3.Pdf

S.Morasca, L. C. B. Property-Based Software: Softw. Eng. Ieee Trans, 1996.

Sonarqube. Sonarqube. [En Línea], 2019. [Consultado El 20/02/2019]. Disponible En: [Http://Www.Sonarqube.Org].

Valenciano, J. Auditoría [De] Mantenibilidad [Para] Aplicaciones Según La Iso/Iec 25000, 2015.

Vega León, M. Análisis De La Implantación De La Metodología Scrum Y La Plataforma Tfs En La Gestión De Un Proyecto Con Integración Continua En La Empresa Animsa, 2020.

Velásquez Rojas, J. V., & Villar Alvarez, A. Impacto De La Implementación De Factores De Mantenibilidad Del Software En La Aplicación Web Ronvel-Rent De La Empresa Ronvel Sac, 2020.

Contribuciones de los autores

1. Conceptualización: Lisandra Tamayo Espinosa
2. Curación de datos: -
3. Análisis formal: Lisandra Tamayo Espinosa
4. Adquisición de fondos: -
5. Investigación: Lisandra Tamayo Espinosa
6. Metodología: -
7. Administración del proyecto: Lisandra Tamayo Espinosa, Nemury Silega Martínez
8. Recursos: Lisandra Tamayo Espinosa
9. Software: Lisandra Tamayo Espinosa
10. Supervisión: Lisandra Tamayo Espinosa, Nemury Silega Martínez
11. Validación: Lisandra Tamayo Espinosa

12. Visualización: Lisandra Tamayo Espinosa
13. Redacción – borrador original: Lisandra Tamayo Espinosa
14. Redacción – revisión y edición: Nemury Silega Martínez