ARTÍCULO DE REVISIÓN / REVIEW ARTICLE
https://dx.doi.org/10.14482/inde.39.1.005.3

Desarrollo guiado por comportamiento: buenas prácticas para la calidad de *software*

Behavior Driven Development: Best Practices for Software Quality

Aldo Emanuel Soraluz Soraluz*
Miguel Ángel Valles Coral**
Danny Lévano Rodríguez***

*Estudiante de Ingeniería de Sistemas, Universidad Peruana Unión, Perú. aldosoraluz@upeu.edu.pe. Orcid: https://orcid.org/0000-0001-5285-8496

**Docente. Doctor en Gestión Pública y Gobernabilidad, Universidad Peruana Unión, Perú miguel.valles@upeu.edu.pe. Orcid: https://orcid.org/0000-0002-8806-2892

***Docente. Magíster en Ingeniería de Sistemas, Universidad Peruana Unión, Perú danlev@upeu.edu.pe. Orcid: https://orcid.org/0000-0002-1783-1105

Correspondencia: Aldo Emanuel Soraluz Soraluz, Estudiante, Universidad Peruana Unión, Perú Teléfono: +51 980292427, Dirección: Calle Condamine #700, Yurimaguas-Perú. aldosoraluz@upeu.edu.pe

Resumen

Asegurar la calidad y funcionalidad de un producto de *software* es garantizar su correcta estructura, composición, ejecución e integridad, pero en algunos casos estas características se ven afectadas debido a la ineficiente gestión y desarrollo del *software*. El objetivo de la revisión fue identificar buenas prácticas al usar el desarrollo guiado por comportamientos. Para su desarrollo, se indagó en artículos de investigación categorizados en revistas indexadas en bases de datos como IEEE, ScienceDirect, Scielo, Scopus y Redalyc entre 2016 y 2020. El análisis y la revisión permitieron identificar buenas prácticas como el uso de los escenarios solo para pruebas de funcionalidad, organizar las características en carpetas de acuerdo con los escenarios del sistema, contextualizar el funcionamiento al mismo idioma de los clientes para facilitar la comunicación, el uso de etiquetas para agrupar escenarios, organizar características según necesidades y generar escenarios sin dependencia. Se concluyó que estas buenas prácticas permiten una correcta comunicación, diseño estructurado del *software*, calidad funcional de cada componente de código y, sobre todo, un producto eficiente con riesgo mínimo de pérdida de recursos y alto margen de éxito.

Palabras clave: calidad de *software*, comportamiento, desarrollo, pruebas.

Abstract

Ensuring the quality and functionality of a software product is to guarantee its correct structure, composition, execution, and integrity, but, in some cases, these characteristics are affected due to inefficient software management and development. The aim of this review was to identify good practices when using behavior-driven development. In its development, we investigated research articles categorized in indexed journals, in databases such as IEEE, ScienceDirect, Scielo, Scopus, and Redalyc, prepared between 2016 and 2020. The analysis and review allowed to identify good practices, such as the use of scenarios that are exclusively for tests of functionality; the organization of characteristics in folders, according to system scenarios; the contextualization of the operation in the same language as the clients, to facilitate communication; the use of labels to group scenarios; the organization of characteristics according to needs; and the generation of scenarios without dependency. We concluded that these good practices allow for adequate communication, structured software design, functional quality of each component of the code, and, above all, for an efficient product with a minimum risk of loss of resources and a high margin of success.

Keywords: behavior, development, software quality, testing.

Fecha de recepción: 26 de octubre de 2020
Fecha de aceptación: 8 de junio de 2021

1. INTRODUCCIÓN

Un estudio realizado en 2019 por [1] en Europa muestra que el desarrollo de *software* centra un 59,4 % de esfuerzo en el uso de pruebas de *software* para asegurar la calidad. Este es el factor principal considerado por las empresas para el rendimiento y la mejora continua de la industria de desarrollo. La implementación de pruebas de *software* en el desarrollo de productos de *software* según [2] y [3] es considerada como componente esencial para asegurar su estructura funcional y de calidad, reconocido también como herramienta útil que proporciona información detallada del cumplimiento de requerimientos del tipo funcional, no funcional, estructural y no estructural en la composición de un producto de *software* durante todo su ciclo de vida.

Según [4] el desarrollo guiado por comportamiento o *behaviour-driven development* (BDD) es una evolución del desarrollo guiado por pruebas o *test-driven development* (TDD) con un proceso más efectivo y visto como la nueva metodología ágil de segunda generación basada en automatización de pruebas, con herramientas y procesos de análisis de mejor composición y desempeño del *software*, que ayuda a prevenir diversos problemas y garantiza la calidad funcional.

Del mismo modo [5] sostienen que es un conjunto de pruebas y técnicas basadas en métricas para evaluar el rendimiento y los atributos de calidad relacionados con un sistema. Algunos análisis con estudios de [6] demostraron que entre el 25 y el 90 % del presupuesto de desarrollo de *software* es requerido en la fase de pruebas y correcciones debido a la ineficiencia e ineficacia del proceso de desarrollo, y que con el uso correcto de pruebas de *software* se disminuye la cantidad de posibles errores humanos y el consumo de recursos, y como resultado se obtiene la reducción del costo en el proyecto de elaboración del *software*.

Dado que la industria de desarrollo tiene un alto margen de crecimiento evolutivo año tras año y la calidad en su estructura es un detalle fundamental en todo el mundo según [7] y [8], se necesita una herramienta como BDD que nos ayude a combatir debilidades en el desarrollo de *software* y nos permita sacar el mejor provecho a su composición con la mínima tasa de error y con un enorme alcance funcional de toda la estructura de manera que las empresas de desarrollo tengan éxito en sus proyectos de *software*.

Perú, en 2019, tuvo un ingreso aproximado de US$800 millones en servicios, del cual el 37 % era del rubro moderno, es decir, de la exportación de tecnología como productos de *software* [9], un indicador que el país tiene crecimiento en ese rubro y, por tanto, genera oportunidades para que el sector tecnológico peruano pueda exportar productos de *software*, entre otras tecnologías. Según [10] la situación tecnológica peruana ha mejorado, por lo que es necesario garantizar que los productos tecnológicos desarrollados sean eficientes, óptimos y de calidad.

Asimismo [11] indicó que Perú experimenta una transformación con respecto a servicios digitales ante empresas y la sociedad, con una tasa de incremento anual del 14,1 %, con visión de multiplicarse 2,5 veces en 2020 ante el crecimiento que tuvo en 2010. Asegurar calidad estructural y evolución a través de la mejora continua es un argumento puesto en cuestionamiento debido a que en Perú solo el 7 % de las empresas manejan buenas prácticas en el desarrollo de *software* según un estudio realizado por [12].

En razón de lo expuesto, esta revisión bibliográfica tuvo como objetivo analizar en profundidad las buenas prácticas del desarrollo basado en el comportamiento en las diferentes etapas de un proyecto de *software* y su contribución para generar un producto de *software* de calidad, en atención a las ilustraciones de documentos experimentales y de revisión, publicados en los últimos cinco años, vinculados a los distintos tipos de pruebas de desarrollo y metodologías para la construcción de *software* con el fin de obtener una síntesis clara del método.

2. METODOLOGÍA

Para alcanzar el objetivo planteado, y dar respuesta al cuestionamiento ¿qué buenas prácticas tiene el desarrollo basado en comportamiento para asegurar la calidad del *software*?, se realizó un proceso exhaustivo de aprendizaje y búsqueda para seleccionar la información que forma parte de esta revisión.

Fase 1: Metacognición

Para el desarrollo de la revisión, se realizó un análisis profundo del tema a través del metaaprendizaje, aplicando los principios de la definición investigativa de [13], que referencia hacer uso de metacognición para la correcta identificación de ideas principales, contexto y palabras clave con el fin de tener una síntesis clara y objetiva, las mismas que posteriormente fueron usadas en el desarrollo de esta revisión.

Fase 2: Búsqueda de información

De acuerdo con el estudio de [14] sobre planteamiento de estrategias para la efectividad y eficiencia en investigación, se realizó la búsqueda de datos e información fundamental, se usó la metodología de revisión sistemática de la literatura y se destacaron aquellos artículos escritos en un lenguaje formal-científico tanto en español como en inglés y técnicas de búsqueda para la correcta selección de información de las distintas fuentes bibliográficas, tales como:

- Búsqueda en bases de datos con mayor realce en calidad de indexación, citación y contenido: IEEE, ScienceDirect, Scielo, Scopus y Redalyc.

- Uso de palabras clave: "behavior driven development", "software quality", "development testing", "software testing" en inglés, y "pruebas unitarias", "pruebas de *software*", "calidad de *software*", "automatización de pruebas" y "desarrollo de *software*" en español.

Asimismo, se utilizó el filtro de búsqueda periódica anual desde 2016 hasta 2020 y se recopiló información actual y trascendental.

La información obtenida fue alojada en el *software* Mendeley en sus plataformas web y de escritorio, con información relevante de los documentos, de donde se dio lectura y la extracción de ideas y fuentes contextuales de autores seleccionados para la construcción de esta revisión.

Fase 3: Análisis de la información

De acuerdo con las investigaciones y guías de [15] y [16], y su sugerencia de aplicar el desarrollo estructurado de la revisión, se llevó a cabo mediante el análisis de la información obtenida de los artículos científicos, estructurada de manera jerárquica desde la definición como parte principal, seguido de la estructura del tema y, por último, la utilidad a través de buenas prácticas, diseñado estratégicamente para sintetizar de forma correcta la información obtenida.

DESARROLLO DE LA REVISIÓN

[17] sostiene que los productos de *software* son herramientas que ayudan a que los procesos diarios se den de forma óptima y automatizada, y así disminuir tiempo, costo y recursos en su ejecución. Parte fundamental de los mencionados productos es asegurar su calidad, que de acuerdo con [18], se encuentran expuestos en cualquier etapa del desarrollo del *software* e, incluso, hasta luego de su despliegue en producción; por ello, indica que es importante conocer los fallos que se van dando y a su vez crear estrategias de reducción de riesgos con la finalidad de asegurar la calidad funcional del *software*.

Según estudios realizados por [19] y [20], asegurar la calidad del *software* conlleva diversos métodos, metodologías, herramientas y técnicas que permiten gestionar la calidad del producto; aunque implique mayor tiempo y presupuesto, es la mejor estrategia para desarrollar *softwares* de calidad totalmente funcionales con el mínimo porcentaje de exposición a fallas. Por ende [21], indica que los procesos como el desarrollo guiado por BDD influyen en el acompañamiento y en la documentación durante el proceso de desarrollo, y de esta manera permite prevenir y verificar fallas, así como garantizar mejora continua a través del autoaprendizaje de errores.

Desarrollo guiado por comportamiento

El BDD es un proceso de desarrollo de *software* ágil de segundo plano, que congenia con diversas metodologías ágiles, en especial, en la frecuencia de programación extrema para el desarrollo de *software* y microservicios, usa el lenguaje específico de dominio o *domain-specific language* (DSL) para empresas denominado Gherkin, que describe el comportamiento del *software* de acuerdo con sus funcionalidades, como manifiesta [4]. En otras palabras, [5] lo describe como una descripción del comportamiento del *software* en lenguaje natural en lugar del uso de un lenguaje de programación.

BDD, al ser un proceso de testeo y planificación de pruebas, es considerado como una herramienta de integración continua y de detección de fallos, que, de acuerdo con el estudio de [22] y [23], estas características permiten alinear el producto de *software* hacia el aseguramiento de la calidad, en atención a diversas evaluaciones, como aceptabilidad, medición, conformidad y estructura en distintas fases y partes del desarrollo. Por otro lado, [24] y [25] resaltan las características que toma un equipo al emplear desarrollo basado en comportamiento, como gestión de oportunidades, orientación de *performance*, contribuciones, preferencias y revisión de objetivos, que permite seguimiento y mejora continua.

[26] y [27] indican que, aunque el mayor esfuerzo de BDD sea en la fase de construcción y pruebas, su enfoque se puede aplicar en diversas partes del desarrollo de *software*. En la fase de planificación, para entender el comportamiento del negocio; en la fase de análisis, para evaluar el comportamiento de negocio y desglosarlas en capturas del comportamiento para el sistema, y en la fase de implementación, en que se llevan a cabo las pruebas de aceptación. Todas estas fases son derivadas de diversos escenarios en diferentes etapas, que siguen un conjunto de reglas de mapeo, del cual se obtiene un pensamiento dirigido por comportamientos para desarrolladores, que facilita visualización de desarrollo, roles, responsabilidades y diversos objetos que ayudan a tener un producto de *software* más estructurado y dirigido.

Los conceptos mencionados por [4], [5] y [26] detallan que BDD es un proceso que puede ser usado con distintos tipos de metodologías para el desarrollo de *software* en sus diferentes etapas, y su importancia recae en dirigir correctamente la construcción del *software* realizando mapeos y mejoras a través de pruebas detalladas de la estructura y componentes, que obtiene como resultado de su implementación una documentación detallada en un lenguaje comprensible como es Gherkin, el cual usa composición lógica de programación que tiene el *software* y da como resultado documentación sintetizada basada en funcionalidades, que [28] menciona como un refinamiento de buenas prácticas para asegurar la calidad del *software*.

Estructura del desarrollo guiado por comportamiento

[4] y [28] representan la estructura de BDD a través de un flujograma de procesos, que incluye una sección reiterativa de TDD.

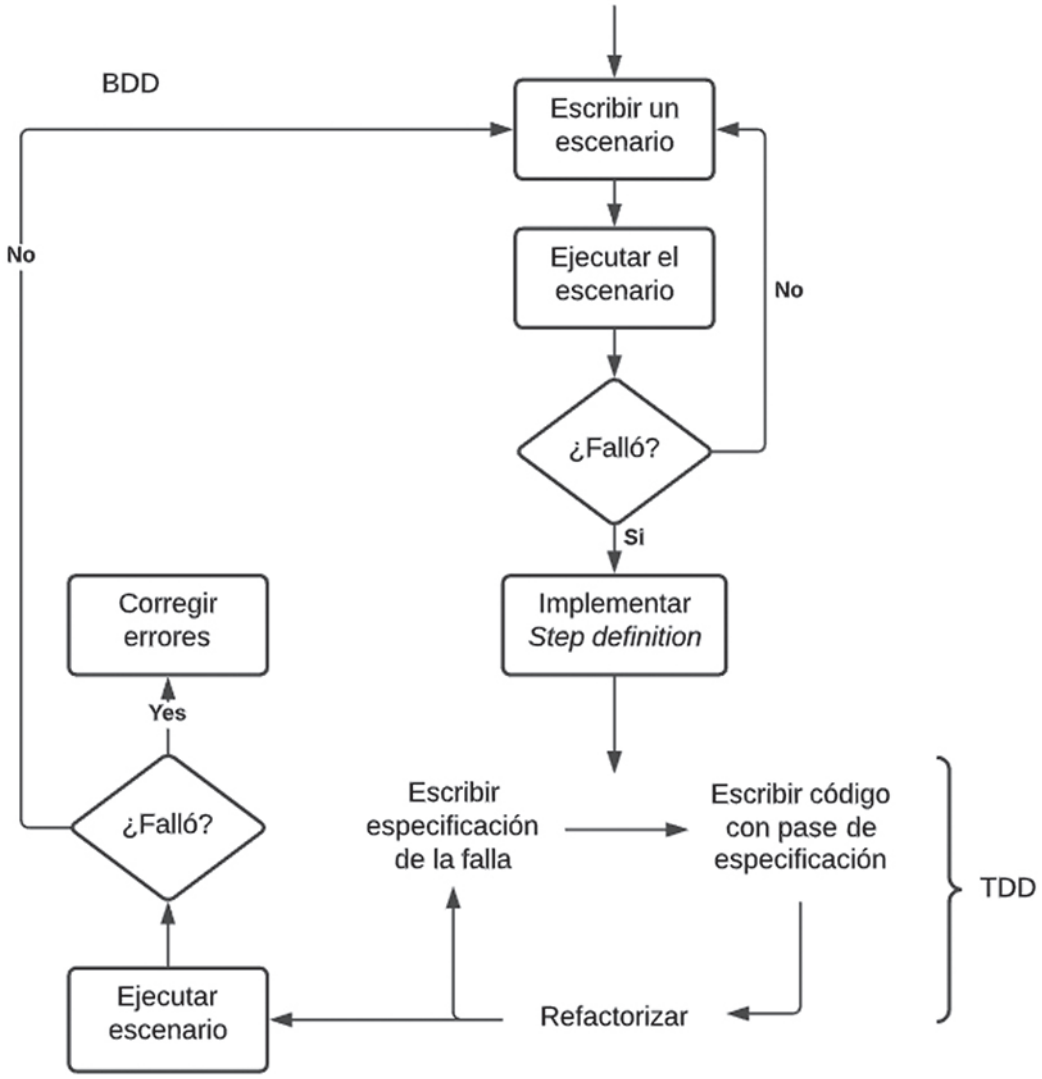

Figura 1. Estructura del desarrollo guiado por comportamiento [4]

Por otro lado, las investigaciones de [5], [29] y [30] tienen una representación distinta y detallada de BDD basado en las pruebas de carga impulsadas por el comportamiento o *behavior-driven load testing* (BDLT), colocando, en primera instancia, el escenario donde se probarán las diferentes características, desglosadas en tres especifica-

ciones: Dado (Given) donde van las precondiciones del suceso o la situación a probar, Cuando (When) que define la acción que se realizará ante dicha situación y Entonces (Then) que es el resultado esperado de la acción ejecutada.

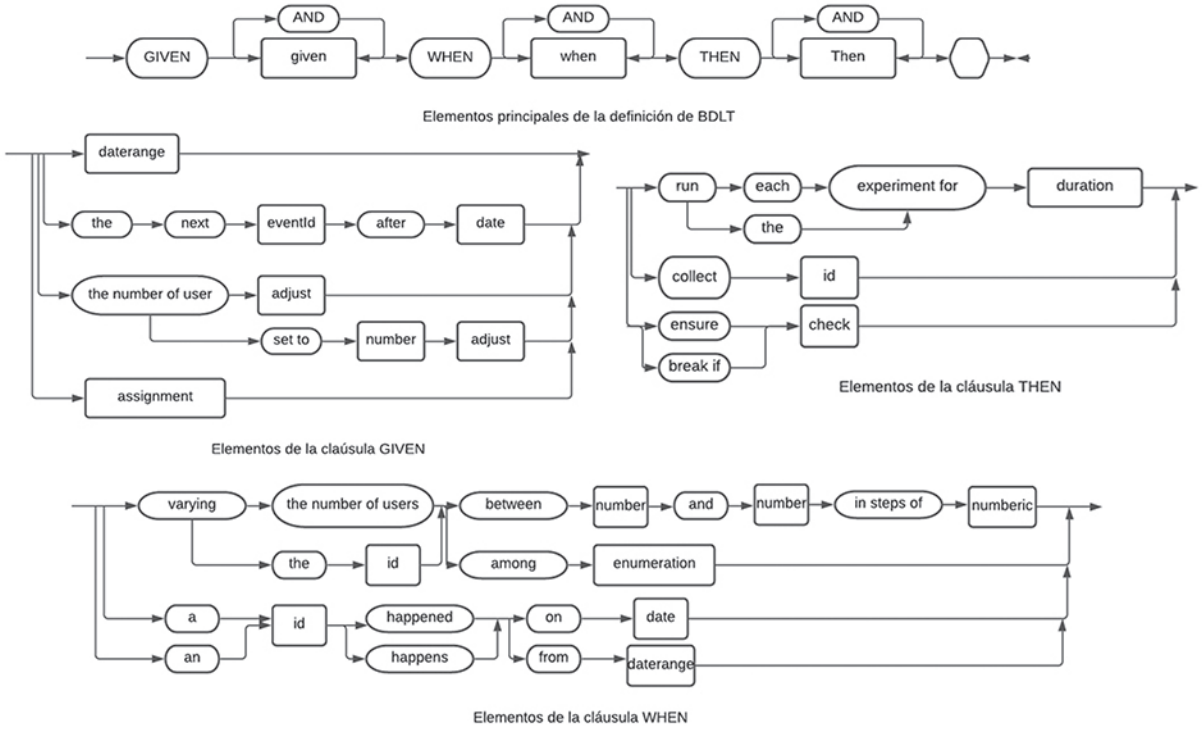

Figura 2. Estructura del lenguaje de BDLT [5]

De acuerdo con la observación de las figuras 1 y 2, BDD se basa en un análisis a través de escenarios dada la característica o funcionalidad del sistema, que cuenta con tres componentes principales: *given, when* y *then,* que prácticamente es la especificación en detalle de una acción dada en un suceso con un resultado esperado, el cual representa cierta funcionalidad del sistema.

Buenas prácticas del desarrollo guiado por comportamiento

Como afirma [31] los escenarios siempre prueban una funcionalidad, son representaciones de historias de usuario en que es necesaria la especificación en detalle de cada componente y método funcional con palabras clave. [32] y [33] argumentan que el uso de Gherkin permite obtener documentación en 60 lenguajes bajo sintaxis descriptiva del comportamiento. Asimismo [34] y [35] expresan que la prueba de escenarios nos facilita conocer ampliamente cada parte funcional del sistema con la posibilidad de

observar errores mediante pruebas dirigidas, y así realizar la corrección respectiva de manera anticipada.

Organización de características por carpetas de acuerdo con los escenarios

El conjunto de escenarios son la representación de una característica de nuestro sistema, por tanto, su organización es fundamental. Según [36] y [37] el proceso paulatino de etapas y pruebas del *software* tiene como resultado la producción de documentación amplia, de diversas funcionalidades, por lo cual es importante su clasificación, no solo para mantener una estructura, sino para ubicar ciertos parámetros de medición para su análisis y aprobación.

Hablar el mismo idioma de los clientes para facilitar la comunicación

[26] y [38] indican que la comunicación con el cliente es vital para la claridad de los requerimientos, debido a que el producto está basado en sus necesidades; BDD con el lenguaje Gherkin permite el diálogo mediante comunicación basada en comportamiento, de manera que guíe y encamine el desarrollo acorde con lo que el cliente solicite. [39] y [40] afirman que la estructura de BDD es entendible ante cualquier tipo de cliente por su lenguaje de representación simple y detallada que a su vez muestra el objetivo y la situación.

Uso de etiquetas en la división de características y escenarios

En diversos proyectos, la estructura interna del código es una característica problemática, ya que algunas funcionalidades representan duplicidad e, incluso, algunas podrían ser parecidas pero usadas para diferentes propósitos. [25] resalta en su proyecto de modelamiento que el uso de etiquetas es fundamental para identificar problemas en el código o duplicidad de métodos, lo cual es respaldado por [41] que indica que estas acciones ayudan a evitar errores como borrar una funcionalidad que parece ser duplicada cuando el propósito es otro, asimismo, generar seguridad, consistencia, eficiencia y calidad en el producto final.

Escenarios independientes para no generar dependencia

La dependencia de funcionalidades es el factor principal por el que los sistemas empiezan a generar fallas según el análisis realizado por [40]. Asimismo, según la investigación de proyección a cambios de [42], indican que la independencia de escenarios es un punto favorable que permite trabajar de manera más ágil el desarrollo del proyecto y realizar correcciones puntuales de fallas, sin afectar la estructura completa o característica del *software*.

[43] manifiesta en su experimento de integrar BDD en un sistema que genera códigos de prueba a partir de diagramas de problema que el desarrollo basado en comportamiento le sirvió como facilitador del entendimiento general de este y le ofreció una guía intuitiva a las partes interesadas para su respectiva comprensión y solución a los problemas planteados de manera gráfica.

Del mismo modo [44] y [45] indican que su experimentación en la prueba de calidad de los escenarios y las buenas prácticas de la estructura BDD son una representación concisa, confiable, negociable, priorizada, comprobable, pequeña, comprensible, inequívoca y valiosa de todas las características funcionales del *software*, categorizándolo como una metodología fundamental para el desarrollo de la calidad estructural, funcional y documental de un producto de *software*. De modo similar a los estudios de [46] y [47], que, aplicando BDD en proyectos de código abierto, resaltan que la integración es beneficiosa para cualquier tipo de metodología, ya que al ser flexible permite aprovechar lo mejor de diversas otras metodologías de manera integrada.

Cada una de las buenas prácticas analizadas conllevan esfuerzo y especialización en el lenguaje Gherkin, tal y como lo experimenta [48] al aplicar BDD en el desarrollo de aplicaciones móviles para reducir errores. Indica que, siendo un lenguaje óptimo de entendimiento, necesita ser realizado con precaución y detalle, de tal forma que permita diferenciarse espacios de códigos, métodos y funcionalidades en diferentes escenarios, para caracterizar el comportamiento del *software* y asegurar el entendimiento claro del usuario final a través de comunicación directa y lenguaje natural.

En función de las necesidades o funcionalidades que se presentan, se pueden crear escenarios que pueden ser reutilizados, y así permitir ahorrar tiempo en la fase del prediseño o reestructuración de las nuevas pruebas, tomar esto como mejora continua según experiencias y establecer estándares según valores de calidad que permitan medir lo apto y óptimo del código testeado.

Asimismo, los hitos de desarrollo pueden ser referentes de calidad si se incluyen pautas de BDD como el uso de su estructura que intensifica el reconocimiento de errores, mejora la categorización de segmentos de código, organiza el desarrollo del proyecto y ayuda a que el cliente de *software* también sea partícipe opinando desde su perspectiva en un lenguaje entendible.

CONCLUSIONES

Las buenas prácticas son el referente a seguir para alcanzar el éxito en la práctica de esta metodología. Con esta revisión, se pudo evidenciar que el desarrollo guiado por comportamiento, su composición y estructura permite llegar de manera eficiente a

los clientes o interesados, es decir, que la comunicación sea fluida con un lenguaje entendible empresarial como Gherkin. Asimismo, las pautas dadas facilitan estructurar el código y reconocer sus funciones, características y descripciones.

Las buenas prácticas identificadas son uso de etiquetas en la construcción del código, empleo de escenarios para el diseño de las funcionalidades del *software*, agrupación de los escenarios en carpetas de acuerdo con sus características y hablar el mismo idioma con los clientes para facilitar la comunicación.

Seguir el modelo BDD sería sinónimo de asegurar la calidad funcional de un producto de *software* con una tasa mínima de errores y un porcentaje alto de funcionalidad total. Buenos conceptos como generar componentes sin dependencia y reutilizar código son bien recibidos en este modelo, con el cual se podrá dirigir en detalle el correcto desarrollo de *software* y alcanzar la madurez de trabajo personal como también del producto.

REFERENCIAS

[1] M. Kuhrmann, P. Diebold, J. Münch, P. Tell, K. Trektere and F. McCaffery, V. Garousi, M. Felderer, O. Linssen, E. Hanser y C. R. Prause, "Hybrid software development approaches in practice: a European perspective", *IEEE Software*, vol. 36, no. 4, pp. 20-31, en. 2018. Doi: 10.1109/MS.2018.110161245

[2] H. Bünder y H. Kuchen, "A model-driven approach for behavior-driven GUI testing", en *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 2019, pp. 1742-1751. https://doi.org/10.1145/3297280.3297450

[3] M. Callejas-Cuervo, A. C. Alarcón-Aldana y A. M. Álvarez-Carreño, "Modelos de calidad del *software*, un estado del arte", *Entramado*, vol. 13, no. 1, pp. 236-250, en.-jun. 2017. https://doi.org/10.18041/entramado.2017v13n1.25125

[4] A. S. Dookhun y L. Nagowah, "Assessing the effectiveness of test-driven development and behavior-driven development in an industry setting", en *2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*, 2019, pp. 365-370. Doi: 10.1109/ICCIKE47802.2019.9004328

[5] H. Schulz, D. Okanović, A. Van Hoorn, V. Ferme y K. S. P. A. Switzerland, "Behavior-driven load testing using contextual knowledge: approach and experiences", en *ICPE'19: Tenth ACM/SPEC International Conference on Performance Engineering*, 2019, pp. 265-272. https://doi.org/10.1145/3297663.3309674

[6] A. C. Barus, "The implementation of ATDD and BDD from Testing Perspectives", *J. Phys.: Conf. Ser.*, vol. 1175, no. 012112, pp. 1-7, 2019. Doi: 10.1088/1742-6596/1175/1/012112

[7] T. R. Benala y R. Mall, "DABE: differential evolution in analogy-based software development effort estimation", *Swarm Evol. Comput.*, vol. 38, pp. 158-172, febr. 2018. https://doi.org/10.1016/j.swevo.2017.07.009

[8] A. Dávila, C. García y S. Cóndor, "Análisis exploratorio en la adopción de prácticas de pruebas de *software* de la ISO/IEC 29119-2 en organizaciones de Lima, Perú", *RISTI - Rev. Iber. Sist. e Tecnol. Inf.*, no. 21, pp. 1-17, mzo. 2017. Doi: 10.17013/risti.21.1-17

[9] *El Peruano*, "Impulso a exportación de servicios", sept. 2019 [En línea]. Disponible en: https://www.datasur.com/impulso-a-exportacion-de-servicios/

[10] *Gestión*, "Perú tiene potencial para desarrollo de *software* dedicado al sector empresarial", abr. 2016 [En línea]. Disponible en: https://gestion.pe/tecnologia/peru-potencial-desarrollo-software-dedicado-sector-empresarial-117244-noticia/

[11] *Gestión*, "Mercado de la informática en Perú crecerá 9.7 % este año", mzo. 2019 [En línea]. Disponible en: https://gestion.pe/economia/empresas/mercado-informatica-peru-crecera-9-7-ano-260535-noticia/

[12] D. E. Muñoz (2019), *Programa Crea Software Perú* [En línea]. Disponible en: https://www.academia.edu/8312872/PROGRAMA_CREA_SOFTWARE_PERU

[13] M. Ángel Valenzuela, " ¿Qué hay de nuevo en la metacognición? Revisión del concepto, sus componentes y términos afines", *Educ. e Pesqui.*, vol. 45, e187571, 2019. https://doi.org/10.1590/S1678-4634201945187571

[14] F. Arias, "Efectividad y eficiencia de la investigación tecnológica en la universidad", *Rev. Electrónica Cienc. y Tecnol. del Inst. Univ. Tecnol. Maracaibo*, vol. 3, no. 1, pp. 64-84, 2017 [En línea]. Disponible en: http://www.recitiutm.iutm.edu.ve/index.php/recitiutm

[15] M. Schwarz-Díaz (febr. 2017), *Guía de referencia para la elaboración de una investigación aplicada* [En línea]. Disponible en: https://core.ac.uk/download/pdf/162614981.pdf

[16] E. M., Sánchez, H. T. Bózzola, A. Soler y S. I. Mariño, "Metodología para el relevamiento y análisis de la información", *Cienc. Lat. Rev. Científica Multidiscip.*, vol. 4, no. 1, pp. 99-115, 2020. https://doi.org/10.37811/cl_rcm.v4i1.44

[17] A. Rodrigues da Silva, C. R. Paiva y V. E. R. da Silva, "Towards a test specification language for information systems: focus on data entity and state machine tests", en *Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development - MODELSWARD*, 2018, pp. 213-224. Doi: 10.5220/0006608002130224

[18] J. Mera Paz, "Análisis del proceso de pruebas de calidad de software", *Ing. Solidar.*, vol. 12, no. 20, pp. 163-176, oct. 2016. http://dx.doi.org/10.16925/in.v12i20.1482

[19] D. Carrizo y A. Alfaro, "Método de aseguramiento de la calidad en una metodología de desarrollo de *software*: un enfoque práctico", *Ingeniare. Rev. chil. ing.*, vol. 26, no. 1, pp. 114-129, mzo. 2018. http://dx.doi.org/10.4067/S0718-33052018000100114

[20] F. Sambinelli, E. L. Ursini, M. A. F. Borges y P. S. Martins, "Modeling and performance analysis of scrumban with test-driven development using discrete event and fuzzy logic", en *2018 6th International Conference in Software Engineering Research and Innovation (CONISOFT)*, 2018, pp. 152-159. Doi: 10.1109/CONISOFT.2018.8645924

[21] R. K. Lenka, S. Kumar y S. Mamgain, "Behavior driven development: tools and challenges", en *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, 2018, pp. 1032-1037. Doi: 10.1109/ICACCCN.2018.8748595

[22] S. Ibarra y M. Muñoz, "Support tool for software quality assurance in software development", en *2018 7th International Conference On Software Process Improvement (CIMPS)*, 2018, pp. 13-19. Doi: 10.1109/CIMPS.2018.8625617

[23] T. Rocha, P. Borba y J. P. Santos, "Using acceptance tests to predict files changed by programming tasks", *J. Syst. Softw.*, vol. 154, pp. 176-195, ag. 2019. https://doi.org/10.1016/j.jss.2019.04.060

[24] R. Kumar, N. Hasteer y J.-P. Van Belle, "Evaluating factors influencing contestant behavior in competitive software development", en *2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 2018, pp. 20-25. Doi: 10.1109/CONFLUENCE.2018.8442860

[25] D. Lubke y T. van Lessen, "Modeling test cases in BPMN for behavior-driven development", *IEEE Softw.*, vol. 33, no. 5, pp. 15-21, 2016. Doi: 10.1109/MS.2016.117

[26] M. G. Cavalcante y J. I. Sales, "The behavior driven development applied to the software quality test: a case study applied to the promotion of sports financing in Brazil", en *2019 14th Iberian Conference on Information Systems and Technologies (CISTI)*, 2019, pp. 1-4. Doi: 10.23919/CISTI.2019.8760965

[27] V. T. Sarinho, "'BDD Assemble!': A paper-based game proposal for behavior driven development design learning", en *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11863 LNCS, pp. 431-435, 2019. https://doi.org/10.1007/978-3-030-34644-7_41

[28] M. M. Moe, "Comparative Study of test-driven development (TDD), behavior-driven development (BDD) and acceptance test-driven development (ATDD)", *Int. J. Trend Sci. Res. Dev.*, vol. 3, no. 4, pp. 231-234, 2019 [En línea]. Disponible en: http://www.ipindexing.com/article/14720

[29] M. O'Brien, "Toward leveraging gherkin controlled natural language and machine translation for global product information development", en *Proceedings of the 21st Annual Conference of the European Association for Machine Translation: 28-30 May 2018*, J. A. Pérez-Ortiz, *et al.*, Eds. Alacant: Universitat d'Alacant, 2018, pp. 293-296 [En línea]. Disponible en: http://hdl.handle.net/10045/76107

[30] S. Staroletov, "Building a process of trustworthy software developing based on BDD and ontology approaches with further formal verification", en *9th Workshop "Program Semantics, Specification and Verification: Theory and Applications" dedicated to the memory of B. A. Trakhtenbrot, M. I. Dekhtyar, and M. K. Valiev (Yaroslavl, Russia, June 21-22, 2018)*, N. Shilov y V. Zakharov, Eds. Yaroslavl: Yaroslavl State University, 2018, pp. 92-97 [En línea]. Disponible en: http://www.lib.uniyar.ac.ru/edocs/iuni/20180406.pdf#page=92

[31] C. Maldonado, I. Gastañaga, C. I. Inchaurrondo, P. A. Vaca, M. Bueno, M. S. Romero y J. P. Peretti, "Estudio de adopción de técnicas de desarrollo de *software* guiado por las pruebas", en *XVIII Work. Investig. en Ciencias la Comput.*, Red de Universidades con Carreras en Informática, Entre Ríos, Argentina, 2016 [En línea]. Disponible en: http://sedici.unlp.edu.ar/handle/10915/53454

[32] I. Herman y M. Plechawska-Wójcik, "Study on applying the Cucumber tool in testing applications", *J. Comput. Sci. Inst.*, vol. 11, pp. 91-95, 2019. https://doi.org/10.35784/jcsi.146

[33] C. Wiecher, S. Japs, L. Kaiser, J. Greenyer, R. Dumitrescu y C. Wolff, "Scenarios in the loop: integrated requirements analysis and automotive system validation", en *MODELS '20: Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, 2020, pp. 1-10. https://doi.org/10.1145/3417990.3421264

[34] A. Sheshasaayee y P. Banumathi, "Impacts of behavioral driven development in the improvement of quality software deliverables", en *2018 3rd International Conference on Inventive Computation Technologies (ICICT)*, 2018, pp. 228-230. Doi: 10.1109/ICICT43934.2018.9034287

[35] R. R. Dania Mailen, Z. Pérez Morales y M. D. Delgado Dapena, "Generador de valores interesantes para casos de pruebas unitarias", *Ing. Ind.*, vol. XL, no. 2, pp. 183-193, my.-ag. 2019 [En línea]. Disponible en: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1815-59362019000200183

[36] G. Lucassen, F. Dalpiaz, J. M. E. M. van der Werf, S. Brinkkemper y Di. Zowghi, "Behavior-driven requirements traceability via automated acceptance tests", en *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, 2017, pp. 431-434. Doi: 10.1109/REW.2017.84

[37] A. Scandaroli, R. Leite, A. H. Kiosia y S. A. Coelho, "Behavior-driven development as an approach to improve software quality and communication across remote business stakeholders, developers and QA: two case studies", en *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)*, 2019, pp. 105-110. Doi: 10.1109/ICGSE.2019.00030

[38] M. Härlin (2016), *Testing and Gherkin in agile projects* [En línea]. Disponible en: https://www.diva-portal.org/smash/record.jsf?pid=diva2 %3A908749&dswid=9136

[39] B. Gonen y D. Sawant, "Significance of agile software development and SQA powered by automation", en *2020 3rd International Conference on Information and Computer Technologies (ICICT)*, 2020, pp. 7-11. Doi: 10.1109/ICICT50521.2020.00009

[40] M. Rahman y J. Gao, "A reusable automated acceptance testing architecture for microservices in behavior-driven development", en *2015 IEEE Symposium on Service-Oriented System Engineering*, 2015, vol. 30, pp. 321-325. Doi: 10.1109/SOSE.2015.55

[41] F. Huang, Y. Wang, Y. Wang y P. Zong, "What software quality characteristics most concern safety-critical domains?", en *Proceedings - 2018 IEEE 18th International Conference on Software Quality, Reliability, and Security Companion, QRS-C 2018*, 2018, pp. 635-636. Doi: 10.1109/QRS-C.2018.00111

[42] A. Z. H. Yang, D. Alencar da Costa y Y. Zou, "Predicting co-changes between functionality specifications and source code in behavior driven development", en *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, 2019, pp. 534-544. Doi: 10.1109/MSR.2019.00080

[43] N. Gao y Z. Li, "Generating testing codes for behavior-driven development from problem diagrams: a tool-based approach", en *2016 IEEE 24th International Requirements Engineering Conference (RE)*, 2016, pp. 399-400. Doi: 10.1109/RE.2016.54

[44] G. Oliveira y S. Marczak, "On the empirical evaluation of BDD scenarios quality: preliminary findings of an empirical study", en *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, 2017, pp. 299-302. Doi: 10.1109/REW.2017.62

[45] S. Bonfanti, A. Gargantini y A. Mashkoor, "Generation of behavior-driven development C++ tests from Abstract State Machine scenarios", *Commun. Comput. Inf. Sci.*, vol. 929, pp. 146-152, oct. 2018. https://doi.org/10.1007/978-3-030-02852-7_13

[46] F. Zampetti, A. Di Sorbo, C. A. Visaggio, G. Canfora y M. Di Penta, "Demystifying the adoption of behavior-driven development in open source projects", *Inf. Softw. Technol.*, vol. 123, p. 106311, jul. 2020. https://doi.org/10.1016/j.infsof.2020.106311

[47] Y. Wang y S. Wagner, "Combining STPA and BDD for safety analysis and verification in agile development: a controlled experiment", en *Agile Processes in Software Engineering and Extreme Programming. XP 2018. Lecture Notes in Business Information Processing*, J. Garbajosa, X. Wang y A. Aguiar. Eds. Cham: Springer, 2018, pp. 37-53. https://doi.org/10.1007/978-3-319-91602-6_3

[48] Z. Ali, "Behavior-driven development as an error-reduction practice for mobile application testing", *IJCSI*, vol. 16, no. 2, , pp. 1-10, mzo. 2019. https://doi.org/10.5281/zenodo.3234110 1