UNIVERSIDAD PERUANA UNIÓN

FACULTAD DE INGENIERÍA Y ARQUITECTURA

Escuela Profesional de Ingeniería de Sistemas

DESARROLLO GUIADO POR COMPORTAMIENTO: BUENAS PRÁCTICAS PARA LA CALIDAD DE SOFTWARE

Trabajo de Investigación para obtener el Grado Académico de Bachiller en Ingeniería de Sistemas

Autor:

Aldo Emanuel Soraluz Soraluz

Asesor:

Mg. Ing. Cristian Wérner García Estrella

Tarapoto, Diciembre

DECLARACIÓN JURADA DE AUTORÍA DEL TRABAJO DE INVESTIGACIÓN

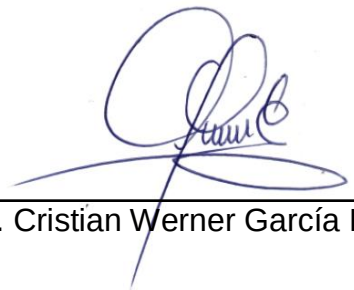
Mg. Ing. Cristian Wérner García Estrella, de la Facultad de Ingeniería y Arquitectura, Escuela Profesional de Ingeniería de Sistemas, de la Universidad Peruana Unión.

DECLARO:

Que la presente investigación titulada: **"DESARROLLO GUIADO POR COMPORTAMIENTO: BUENAS PRÁCTICAS PARA LA CALIDAD DE SOFTWARE"** constituye la memoria que presenta el (la) / los estudiantes(es) Aldo Emanuel Soraluz Soraluz para obtener el Grado Académico de Bachiller en Ingeniería de Sistemas cuyo trabajo de investigación ha sido realizado en la Universidad Peruana Unión bajo mi dirección.

Las opiniones y declaraciones en este informe son de entera responsabilidad del autor, sin comprometer a la institución.

Y estando de acuerdo, firmo la presente declaración en la ciudad de Tarapoto a los 20 días del mes de diciembre del año 2020



Ing. Cristian Werner García Estrella

016

ACTA DE SUSTENTACIÓN DE TRABAJO DE INVESTIGACIÓN

En San Martín, Tarapoto, Morales, a 20 día(s) del mes de diciembre del año 2020 siendo las 11:00 horas se reunieron los miembros del jurado en la Universidad Peruana Unión campus Tarapoto, bajo la dirección del (de la) presidente(a): Mg. Immer Elias Cuellar Rodriguez, el (la) secretario(a): Mg. Joseph Ibrahim Cruz Rodriguez y los demás miembros: Mg. Danny Lévano Rodriguez y el (la) asesor (a) Ing. Cristian Werner Garcia Estrella con el propósito de administrar el acto académico de sustentación del trabajo de investigación titulado: Desarrollo Guiado por comportamiento: buenas prácticas para la calidad de Software.

de los (las) egresados (as): a) Aldo Emanuel Soraluz Soraluz b)

conducente a la obtención del grado académico de Bachiller en Ingeniería de Sistemas
(Denominación del Grado Académico de Bachiller)

El Presidente inició el acto académico de sustentación invitando al candidato (a)/s hacer uso del tiempo determinado para su exposición. Concluida la exposición, el Presidente invitó a los demás miembros del jurado a efectuar las preguntas, y aclaraciones pertinentes, las cuales fueron absueltas por el candidato (a)/s. Luego, se produjo un receso para las deliberaciones y la emisión del dictamen del jurado.

Posteriormente, el jurado procedió a dejar constancia escrita sobre la evaluación en la presente acta, con el dictamen siguiente:

Candidato/a (a): Aldo Emanuel Soraluz Soraluz

CALIFICACIÓN	ESCALAS			Mérito
	Vigesimal	Literal	Cualitativa	
Aprobado	19	A	Excelente	Excelencia

Candidato/a (b):

CALIFICACIÓN	ESCALAS			Mérito
	Vigesimal	Literal	Cualitativa	

(*) Ver parte posterior

Finalmente, el Presidente del jurado invitó al candidato (a)/s a ponerse de pie, para recibir la evaluación final y concluir el acto académico de sustentación procediéndose a registrar las firmas respectivas.

Presidente/a | Secretario/a

Asesor/a | Miembro | Miembro

Candidato/a (a) | Candidato/a (b)

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. 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 elaboradas entre los años 2016 y 2020. El análisis y revisión permitió identificar buenas prácticas, como el uso de los escenarios solo para pruebas de funcionalidad, organizar las características en carpetas de acuerdo a 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. The aim of the review was to identify good practices when using behavior-driven development. In its development, we investigated research articles categorized in journals indexed 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 only for tests of functionality, organize the characteristics in folders according to the system scenarios, contextualize the operation in the same language of the clients to facilitate communication, the use of labels

to group scenarios, organize characteristics according to needs and generate scenarios without dependency. We concluded that these good practices allow adequate communication, structured software design, functional quality of each component of the code and, above all, an efficient product with a minimum risk of loss of resources and a high margin of success.

Keywords: behavior, development, software quality, testing

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, siendo este el factor principal considerado por las empresas para el rendimiento y 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 la estructura funcional y de calidad del mismo, 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] a través de su exploración investigativa, señalan que el desarrollo guiado por comportamiento o *Behavior Driven Development* (BDD por sus siglas en inglés), es una evolución del desarrollo guiado por pruebas o *Test Driven Development* (TDD por sus siglas en inglés) 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 basada en métricas para evaluar el rendimiento y los atributos de calidad relacionados a un sistema, algunos análisis a través de estudios de [6] demostraron que entre el 25% y 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, obteniendo como resultado 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 de 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 el 2019 tuvo un ingreso aproximado de 800 millones de dólares en servicios, del cual 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 lo 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.

Así mismo [11] indicó que Perú experimenta una transformación con respecto a servicios digitales ante empresas y la sociedad, con una tasa de incremento anual de 14.1%, con visión de multiplicarse 2.5 veces en 2020 ante el crecimiento que tuvo en el año 2010; el asegurar calidad estructural y evolución a través de mejora continua es un argumento puesto en cuestionamiento debido a que en Perú solo 7% de 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, teniendo en cuenta ilustraciones de documentos experimentales y de revisión, publicados en los últimos 5 años, vinculado 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 a hacer uso de metacognición para la correcta identificación de ideas principales, contexto y palabras claves con el fin de tener una síntesis clara y objetiva; las mismas que posteriormente fueron usadas en el desarrollo de la presente revisión.

Fase 2: búsqueda de información

De acuerdo al estudio de [14], sobre planteamiento de estrategias para la efectividad y eficiencia en investigación, se realizó la búsqueda de datos e información fundamental haciendo uso de la metodología de revisión sistemática de la literatura, destacando 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 claves: Behavior driven development, software quality, development testing, software testing en el lenguaje inglés y pruebas unitarias, pruebas de software, calidad de software, automatización de pruebas y desarrollo de software en el lenguaje español.

Así mismo se utilizó el filtro de búsqueda periódica anual desde el 2016 hasta el 2020, recopilando 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 la presente revisión.

Fase 3: análisis de la información

De acuerdo a 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.

3 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, disminuyendo así tiempo, costo y recursos en su ejecución; parte fundamental de los mencionados productos es asegurar su calidad, que de acuerdo a la investigación de [18], estos 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], el asegurar la calidad del software conlleva diversos métodos, metodologías, herramientas y técnicas que permiten gestionar la calidad del producto, aunque esto conlleve 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 comportamiento o *Behavior Driven Development* (BDD por sus siglas en inglés) influyen mucho en el acompañamiento y documentación durante el proceso de desarrollo, permitiendo así prevenir y verificar fallas, garantizando mejora continua a través del autoaprendizaje de errores.

Desarrollo guiado por comportamiento

El desarrollo guiado por comportamiento (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 micro servicios, usa el lenguaje específico de dominio o *Domain Specific Language* (DSL por sus siglas en inglés) para empresas denominado Gherkin, que describe el comportamiento del software de acuerdo a sus funcionalidades 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 al estudio de [22] y [23], dichas características permiten alinear el producto de software hacia el aseguramiento de la calidad, teniendo en cuenta 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 es gestión de oportunidades,

orientación de performance, contribuciones, preferencias y revisión de objetivos, permitiendo seguimiento y mejora continua.

Las investigaciones realizadas por [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 el análisis, para evaluar el comportamiento de negocio y desglosarlas en capturas del comportamiento para el sistema, y en la implementación donde se llevan a cabo las pruebas de aceptación; todas estas partes 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 la visualización del desarrollo, roles, responsabilidades y diversos objetos que ayudan a tener un producto de software más estructurado y dirigido.

Los conceptos mencionados por [5], [4] 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, obteniendo como resultados de su implementación una documentación detallada en un lenguaje comprensible como es Gherkin, que usa composición lógica de programación que tiene el software y da como resultado documentación sintetizada basada en funcionalidades, algo que [28] hace mención como un refinamiento de buenas prácticas para asegurad 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, incluyendo una sección reiterativa de TDD:

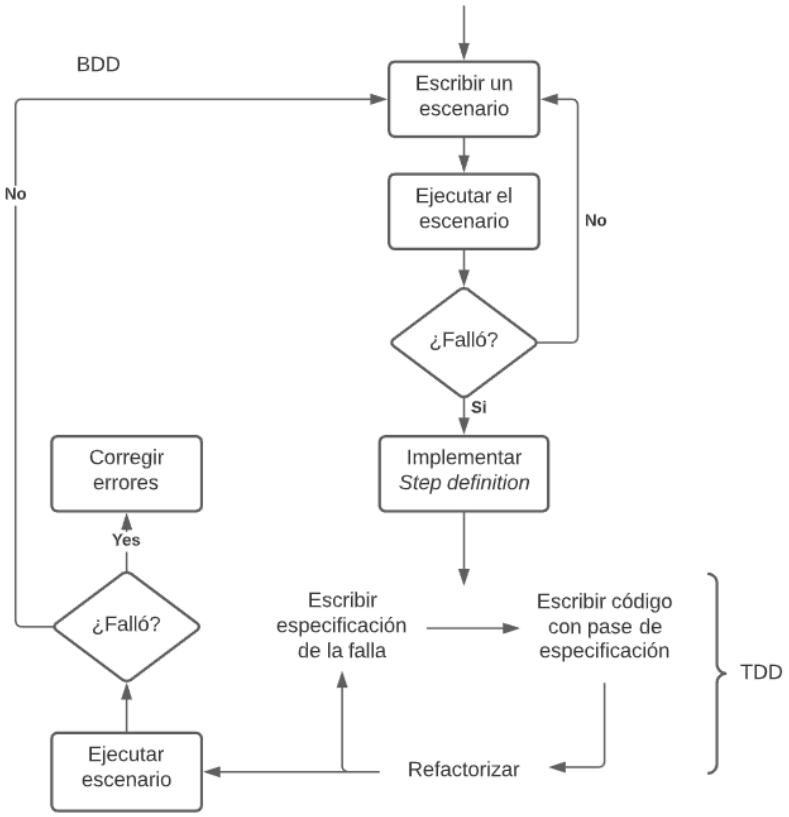


Fig. 1. Estructura del desarrollo guiado por comportamiento. Recuperado de [4]

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 por sus siglas en inglés), colocando en primera instancia el escenario donde se probará las diferentes características, desglosada en 3 especificaciones que son: Dado (Given) donde van las precondiciones del suceso o mejor dicho 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.

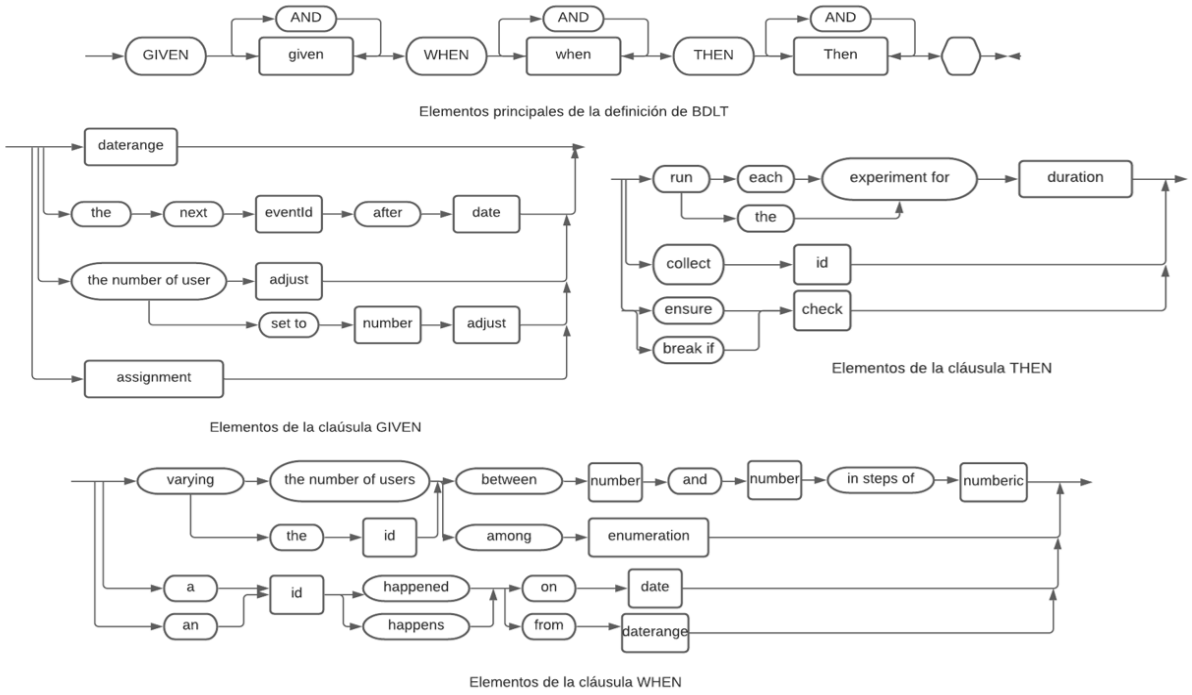


Fig. 2. Estructura del lenguaje de prueba de carga basado en comportamiento (BDLT). Recuperado de [5]

De acuerdo a la observación de las figuras 1 y 2, de ambos autores, BDD se basa en un análisis a través de escenarios dada la característica o funcionalidad del sistema, que cuenta con 3 componentes principales: *given, when,* y *then,* que prácticamente es la especificación a detalle de una acción dada en un suceso con un resultado esperado, que representa cierta funcionalidad del sistema.

3.1 Buenas prácticas del desarrollo guiado por comportamiento

Como afirma [31], los escenarios siempre prueban una funcionalidad, son representaciones de historias de usuario donde es necesaria la especificación a detalle de cada componente y método funcional con palabras claves, [32] y [33] argumentan

que el uso de Gherkin permite obtener documentación en 60 lenguajes diferentes bajo sintaxis descriptiva del comportamiento, así mismo las investigaciones de [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 a los escenarios. El conjunto de escenarios son la representación de una característica de nuestro sistema por lo tanto la organización de los mismo es fundamental, de acuerdo a [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 en sus investigaciones 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 a 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 a su vez, mostrando 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 características podrían ser parecidas pero usadas para diferentes propósitos, [25] hacen énfasis en su proyecto de modelamiento, que el uso de etiquetas es fundamental para identificar problemas en el código o duplicidad de métodos, esto es respaldado por la investigación de [41] que indica que, estas acciones ayudan a evitar errores como el de borrar una funcionalidad que parece ser duplicada cuando el propósito es otro, así mismo 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 empiecen a generar fallas según el análisis realizado por [40]; así mismo 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.

La investigación realizada por [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 del problema, ofreciéndole 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] en su investigación 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, con la finalidad de caracterizar el comportamiento del software, asegurando el entendimiento claro del usuario final a través de comunicación directa y lenguaje natural.

En función a las necesidades o funcionalidades que se presentan, se pueden crear escenarios, que pueden ser reutilizados, permitiendo así ahorrar tiempo en la fase del pre diseño o reestructuración de las nuevas pruebas, tomando esto como mejora continua en base a experiencias y estableciendo estándares en base a valores de calidad que permitan medir lo apto y óptimo del código testeado.

Así mismo 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 participe opinando desde su perspectiva en un lenguaje entendible.

4 CONCLUSIONES

Las buenas prácticas son el referente que se deben seguir para conseguir el éxito en la práctica de esta metodología, a través de la presente revisión se pudo evidenciar que el desarrollo guiado por comportamiento, su composición y estructura permiten llegar de manera eficiente a los clientes o interesados, es decir permite que la comunicación sea fluida a través de un lenguaje entendible empresarial como lo es Gherkin, así mismo 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, uso de escenarios para el diseño de las funcionalidades del software, agrupación de los escenarios en carpetas de acuerdo a sus características y por último hablar el mismo idioma con los clientes para facilitar la comunicación.

Seguir el modelo BDD sería el 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 a detalle el correcto desarrollo de software, alcanzando la madurez de trabajo personal como también del producto.

REFERENCIAS

[1] M. Kuhrmann *et al.*, “Hybrid Software Development Approaches in Practice: A European Perspective,” *IEEE Softw.*, vol. 36, no. 4, pp. 20–31, 2019, doi: 10.1109/MS.2018.110161245.

[2] H. Bünder and H. Kuchen, “A model-driven approach for behavior-driven GUI testing,” in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 2019, pp. 1742–1751, doi: 10.1145/3297280.3297450.

[3] M. Callejas-Cuervo, A. C. Alarcón-Aldana, and A. M. Álvarez-Carreño, “Modelos de calidad del software, un estado del arte,” *ENTRAMADO*, vol. 13, no. 1, pp. 236–250, 2017, doi: 10.18041/entramado.2017v13n1.25125.

[4] A. S. Dookhun and L. Nagowah, “Assessing The Effectiveness Of Test-Driven Development and Behavior-Driven Development in an Industry Setting,” in *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, and K. S. P. A. Switzerland, “Behavior-driven Load Testing Using Contextual Knowledge—Approach and Experiences,” in *ICPE ’19: Tenth ACM/SPEC International Conference on Performance Engineering*, 2019, pp. 265–272, doi: 10.1145/3297663.

[6] A. Christy Barus, “The implementation of ATDD and BDD from Testing Perspectives,” *J. Phys. Conf. Ser.*, vol. 1175, no. 012112, 2019, doi: 10.1088/1742-6596/1175/1/012112.

[7] T. R. Benala and R. Mall, “DABE: Differential evolution in analogy-based software development effort estimation,” *Swarm Evol. Comput.*, vol. 38, pp. 158–172, 2018, doi: 10.1016/j.swevo.2017.07.009.

[8] A. Dávila, C. García, and 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, 2017, doi: 10.17013/risti.21.1-17.

[9] El Peruano, “Impulso a exportación de servicios,” *EL PERUANO*. EL PERUANO, 2019, [Online]. Available: https://elperuano.pe/noticia-impulso-a-exportacion-servicios-83245.aspx.

[10] Gestión, “Tecnología: Perú tiene potencial para desarrollo de software dedicado al sector empresarial,” *GESTIÓN*. NOTICIAS GESTIÓN, 2016, [Online]. Available: https://gestion.pe/tecnologia/peru-potencial-desarrollo-software-dedicado-sector-empresarial-117244-noticia/.

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

[12] D. E. Muñoz, “PROGRAMA CREA SOFTWARE PERU,” 2019. [Online]. Available: http://www.siicex.gob.pe/siicex/resources/sectoresproductivos/ProgramaCREASOFTWAREPERU.pdf.

[13] M. Ángel Valenzuela, “¿Qué hay de nuevo en la metacognición? Revisión del

concepto, sus componentes y términos afines," *Educ. e Pesqui.*, vol. 45, 2019, doi: 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, [Online]. Available: http://www.recitiutm.iutm.edu.ve/index.php/recitiutm.

[15] M. Schwarz, "Guía de referencia para la elaboración de una investigación aplicada," *Repos. Investig. - Univ. Lima*, p. 30, 2017, [Online]. Available: https://core.ac.uk/download/pdf/162614981.pdf.

[16] E. María, Sánchez, H. T. Bózzola, A. Soler, and 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, doi: 10.37811/cl_rcm.v4i1.44.

[17] A. Rodrigues da Silva, A. C. R. Paiva, and V. Emanuel R. da Silva, "Towards a Test Specification Language for Information Systems: Focus on Data Entity and State Machine Tests," *Proc. 6th Int. Conf. Model. Eng. Softw. Dev.*, pp. 213–224, 2018, 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, 2016, doi: 10.16925/in.v12i20.1482.

[19] D. Carrizo and A. Alfaro, "Método de aseguramiento de la calidad en una metodología de desarrollo de software: Un enfoque práctico," *Ingeniare*, vol. 26, no. 1, pp. 114–129, 2018, doi: 10.4067/S0718-33052018000100114.

[20] F. Sambinelli, E. L. Ursini, M. A. F. Borges, and P. S. Martins, “Modeling and Performance Analysis of Scrumban with Test-Driven Development Using Discrete Event and Fuzzy Logic,” in *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, and S. Mamgain, “Behavior Driven Development: Tools and Challenges,” in *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 and M. Munoz, “Support tool for software quality assurance in software development,” in *2018 7th International Conference On Software Process Improvement (CIMPS)*, 2018, pp. 13–19, doi: 10.1109/CIMPS.2018.8625617.

[23] T. Rocha, P. Borba, and J. P. Santos, “Using acceptance tests to predict files changed by programming tasks,” *J. Syst. Softw.*, vol. 154, pp. 176–195, 2019, doi: 10.1016/j.jss.2019.04.060.

[24] R. Kumar, N. Hasteer, and J.-P. Van Belle, “Evaluating Factors Influencing Contestant Behavior in Competitive Software Development,” in *2018 8th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, 2018, pp. 20–25, doi: 10.1109/CONFLUENCE.2018.8442860.

[25] D. Lubke and 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 and 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,” in *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,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2019, vol. 11863 LNCS, pp. 431–435, doi: 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, [Online]. Available: www.ijtsrd.com.

[29] M. O’Brien, “Toward leveraging Gherkin Controlled Natural Language and Machine Translation for Global Product Information Development,” *EAMT 2018 - Proc. 21st Annu. Conf. Eur. Assoc. Mach. Transl.*, pp. 293–296, 2018, [Online]. Available: 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,” pp. 92–97, 2018.

[31] C. Maldonado *et al.*, “Estudio de adopción de técnicas de desarrollo de software guiado por las pruebas,” *XVIII Work. Investig. en Ciencias la Comput.*, pp. 631–636, 2016, [Online]. Available: http://sedici.unlp.edu.ar/handle/10915/53454.

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

[33] C. Wiecher, S. Japs, L. Kaiser, J. Greenyer, R. Dumitrescu, and C. Wolff, “Scenarios in the loop Integrated Requirements Analysis and Automotive System Validation,” *Proc. 23rd ACM/IEEE Int. Conf. Model Driven Eng. Lang. Syst. Companion Proc.*, pp. 1–10, 2020, doi: 10.1145/3417990.3421264.

[34] A. Sheshasaayee and P. Banumathi, “Impacts of behavioral driven development in the improvement of quality software deliverables,” in *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, and M. D. Delgado Dapena, “Generador de valores interesantes para casos de pruebas unitarias,” *Ing. Ind.*, vol. XL, no. 2, pp. 183–193, 2019, [Online]. Available: http://www.rii.cujae.edu.cu.

[36] G. Lucassen, F. Dalpiaz, J. M. E. M. van der Werf, S. Brinkkemper, and Di. Zowghi, “Behavior-Driven Requirements Traceability via Automated Acceptance Tests,” in *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, and 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,” in *2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE)*, 2019, pp. 105–110, doi: 10.1109/ICGSE.2019.00030.

[38] M. Härlin, “Testing and Gherkin in agile projects,” 2016.

[39] B. Gonen and D. Sawant, “Significance of Agile Software Development and SQA Powered by Automation,” in *2020 3rd International Conference on Information and Computer Technologies (ICICT)*, 2020, pp. 7–11, doi: 10.1109/ICICT50521.2020.00009.

[40] M. Rahman and J. Gao, “A Reusable Automated Acceptance Testing Architecture for Microservices in Behavior-Driven Development,” in *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, and P. Zong, “What Software Quality Characteristics Most Concern Safety-Critical Domains?,” in *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, and Y. Zou, “Predicting Co-Changes between Functionality Specifications and Source Code in Behavior Driven Development,” in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, 2019, vol. 2019-May, pp. 534–544, doi:

10.1109/MSR.2019.00080.

[43] N. Gao and Z. Li, “Generating Testing Codes for Behavior-Driven Development from Problem Diagrams: A Tool-Based Approach,” *Proc. - 2016 IEEE 24th Int. Requir. Eng. Conf. RE 2016*, pp. 399–400, 2016, doi: 10.1109/RE.2016.54.

[44] G. Oliveira and S. Marczak, “On the Empirical Evaluation of BDD Scenarios Quality: Preliminary Findings of an Empirical Study,” in *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, 2017, pp. 299–302, doi: 10.1109/REW.2017.62.

[45] S. Bonfanti, A. Gargantini, and A. Mashkoor, “Generation of Behavior-Driven Development C++ Tests from Abstract State Machine Scenarios,” *Commun. Comput. Inf. Sci.*, vol. 929, pp. 146–152, 2018, doi: 10.1007/978-3-030-02852-7_13.

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

[47] Y. Wang and S. Wagner, “Combining STPA and BDD for safety analysis and verification in agile development: A controlled experiment,” *Lecture Notes in Business Information Processing*, vol. 314, Springer Verlag, pp. 37–53, 2018.

[48] Z. Ali, “Desarrollo basado en el comportamiento como una práctica de reducción de errores para pruebas de aplicaciones móviles,” *IJCSI Int. J. Comput. Sci.*

Issues, vol. 16, no. 2, 2019, doi: https://doi.org/10.5281/zenodo.3234110 1.