Propuesta para la automatización de pruebas realizadas en el proceso de aseguramiento de calidad de software en una empresa de desarrollo de software

Yully Catherine Pacanchique Plata

Directora:

Lady Mildred Rojas Galindo

Co-director:

Javier Hernán Jiménez Beltrán

Universidad Nacional Abierta y a Distancia

Escuela de Ciencias Básicas, Tecnología e Ingeniería

Programa de Ingeniería Industrial

Bogotá, Colombia

Agosto de 2024

Declaración de Derechos de Propiedad Intelectual

Los autores de la presente propuesta manifestamos que conocemos el contenido del Acuerdo 06 de 2008, Estatuto de Propiedad Intelectual de la UNAD, Artículo 39 referente a la cesión voluntaria y libre de los derechos de propiedad intelectual de los productos generados a partir de la presente propuesta. Asimismo, conocemos el contenido del Artículo 40 del mismo Acuerdo, relacionado con la autorización de uso del trabajo para fines de consulta y mención en los catálogos bibliográficos de la UNAD.

“El fracaso no es el final, es solo una oportunidad para empezar de nuevo de manera más inteligente”

Grace Hopper

Agradecimientos

Expreso mis agradecimientos a:

A mis padres Elizabeth Plata y Alberto Pacanchique por ser mi fuente constante de apoyo, amor y sabiduría. Su paciencia y aliento fueron la fuerza impulsora detrás de este proyecto. A los docentes de la universidad UNAD por su orientación experta y su compromiso continuo. Cada discusión y consejo fue invaluable para el desarrollo de esta investigación. A mi hermana Diana Pacanchique y mi amigo Andrés López por estar ahí para mí en los altibajos y recordarme constantemente que el trabajo duro y la amistad van de la mano.

Esta investigación está dedicada a todos ustedes cuyas huellas han dejado una huella imborrable en este camino hacia la finalización de mi carrera. Su influencia y apoyo fue la chispa que encendió este logro.

Resumen

El presente documento trata de la evaluación de la eficiencia de la automatización en la productividad del desarrollo de software de una empresa desarrolladora de Software, cuyo nombre se omite, por protección de datos. Actualmente, en la empresa se realizan pruebas funcionales (manuales) para asegurar la calidad del software que se desarrolla. A medida que el proceso se ha expandido, la cobertura de pruebas debe ser más extensa porque es imperativo que en cada salida a producción se garantice la correcta funcionalidad del software existente, así como de los nuevos desarrollos. Esta situación ha dejado en evidencia problemáticas de eficiencia y reprocesos que afectan el nivel cumplimiento en la entrega del servicio, de forma que, es fundamental implementar en el proceso de aseguramiento de la calidad pruebas automatizadas que aumenten el alcance de cobertura de las pruebas en cada publicación a producción y mejore la eficiencia de los tiempos destinados a estas validaciones.

Para instaurar las pruebas automatizadas, es indispensable que la herramienta utilizada sea compatible con el software en proceso de desarrollo, por esta razón, esta propuesta busca identificar la herramienta con mejores beneficios para la automatización del proceso de aseguramiento de calidad de software en la empresa. La metodología para utilizar corresponde a una metodología mixta que contribuirá al reconocimiento y caracterización de las herramientas (lenguajes de programación usados, ventajas y desventajas), y permitirá comprobar la efectividad y el beneficio de las herramientas mediante la valoración numérica (metodología cuantitativa). Como resultado se realizará un documento técnico con la descripción del proceso investigativo y el análisis de las herramientas identificadas, además de las recomendaciones pertinentes de acuerdo con los requerimientos de la empresa.

Palabras claves: Software, Automatización, Pruebas, Calidad

Abstract

This paper is about the evaluation of the efficiency of automation in the productivity of software development of a company whose name is omitted, for data protection. At present, the company conducts functional tests (manual) to ensure the quality of the software being developed. As the process has expanded, testing coverage must be more extensive because it is imperative that each release to production ensures proper functionality of existing software as well as new developments. This situation has left in evidence problems of efficiency and reprocesses that affect the level of compliance in the delivery of the service, It is therefore essential to implement automated tests in the quality assurance process that increase the coverage of the tests in each publication to production and improve the efficiency of the time allocated for these validations.

To implement automated testing, it is essential that the tool used be compatible with the software being developed, for this reason, this proposal seeks to identify the tool with the best benefits for automating the software quality assurance process in the company. The methodology to be used corresponds to a mixed methodology that will contribute to the recognition and characterization of tools (programming languages used, advantages and disadvantages), and will allow to test the effectiveness and benefit of tools by numerical valuation (quantitative methodology). As a result, a technical document will be produced with the description of the investigation process and analysis of the identified tools, in addition to relevant recommendations according to the requirements of the company.

Keywords: Software, Automation, Testing, Quality

Tabla de Contenido

Lista de Figuras

Lista de Tablas

Lista de Gráficas

Lista de Apéndices

Introducción

El desarrollo de este proyecto de grado se realiza debido a la necesidad de evaluar el impacto de implementar automatización de pruebas de software en el ciclo de desarrollo. La importancia de optimizar los procesos de desarrollo de software se hace necesaria para mantener la competitividad en la industria tecnológica. Muchas empresas todavía enfrentan dificultades significativas para incorporar herramientas de automatización que aseguren la eficiencia y la calidad de sus productos.

Realizando que, en el marco de la creciente demanda de soluciones tecnológicas más eficientes, este proyecto de grado propone atender la necesidad de aplicar métodos novedosos de pruebas de software automáticas. De hecho, la necesidad de mejorar la eficiencia de los ciclos de desarrollo de software se ha convertido en un factor crucial para poder competir en el mercado tecnológico. Dicho esto, la mayoría de las organizaciones enfrentan desafíos cruciales en relación con sus esfuerzos por integrar las herramientas de automatización necesarias para garantizar la calidad y eficacia de sus productos.

El objetivo de este proyecto es evaluar la implementación de pruebas automatizadas, reduciendo los tiempos de ejecución y mejorando los procesos operativos. Para ello, se plantean como objetivos específicos diagnosticar el proceso de pruebas actuales, investigación de herramientas para pruebas automatizas e implementación de la mejor opción y evaluación del proceso realizado.

Este proyecto pretende responder a la pregunta ¿Qué impacto puede tener implementar técnicas de automatización del proceso de aseguramiento de calidad en la productividad del desarrollo de software en el marco de los estándares de calidad? La metodología empleada será

Lean Software Development con un enfoque de maximización del valor entregado al cliente y minimizando los desperdicios, adicional combinará análisis cuantitativo de tiempos de prueba con evaluaciones cualitativas del impacto en el proceso del ciclo de desarrollo.

El presente trabajo se estructura en seis capítulos, comenzando con la justificación de este proyecto, seguido de un marco teórico que explora las bases de las pruebas de software, seguido de la metodología usada para el diagnóstico del proceso de pruebas de software, seguido de una revisión de las herramientas existentes para pruebas automatizadas de software, un análisis comparativo, la implementación de las pruebas automatizadas y finalmente conclusiones.

Planteamiento del Problema

La empresa desarrolladora de software que actualmente ofrece el servicio de SAAS para el trámite de facturas y reclamaciones referentes a cobros de las prestaciones de salud realizadas por profesionales e instituciones de salud y remitidas a las diferentes aseguradoras que actualmente son clientes.

Al tener variedad de clientes las funcionalidades del aplicativo deben ser adaptables e incluso únicas según se requiera.

Para la elaboración de software se ha establecido que durante el ciclo que consta de diferentes etapas fundamentales: Planificación, Análisis, Diseño, Implementación, Aseguramiento de Calidad, Despliegue, Utilización y Mantenimiento.

En la etapa aseguramiento de calidad es donde se verifica y valida que lo que se espera se encuentre funcionando y se pueda incorporar nuevos desarrollos. Para la empresa actualmente se realizan pruebas funcionales certificando desde el ingreso hasta la parte final de los diferentes procesos; dentro de estas están las pruebas de caja negra, regresión, humo y usabilidad.

Día tras día el incremento de desarrollos se va ampliando por lo que se hace necesario realizar un mantenimiento de los módulos de la aplicación ya existentes, garantizando al cliente su compatibilidad con los nuevos requerimientos y que su funcionalidad se encuentre intacta.

Las pruebas funcionales manuales requieren establecer nuevas dinámicas, para asegurar una verificación exhaustiva que, a su vez, contribuya a aumentar la efectividad en términos de tiempos de ejecución y que permita al talento humano concentrarse en los flujos críticos y de impacto en cada publicación que se envía a producción.

La problemática se ha logrado identificar a raíz de los siguientes factores: a) el proceso de desarrollo implica estar a la vanguardia para brindar un servicio de calidad, óptimo y oportuno; b) los tiempos que se utilizan para la entrega y los compromisos adquiridos en algunas circunstancias se han visto prolongados debido a la duración de las pruebas requeridas para generar los cambios que pueden afectar las funcionalidades transversales del software; c) actualmente el proceso de verificación del funcionamiento de las aplicaciones antes de la publicación a producción puede tardar entre cinco a siete días; d) adicional, durante la ejecución de las pruebas se descubren errores que, al ser corregidos, dan como resultado una nueva revisión, alargando los tiempos de espera y reprocesamiento.

La motivación detrás de este problema surge de una experiencia anterior donde se estimó incorrectamente el tiempo necesario para entregar un producto mínimo viable a un cliente. A pesar de asignar un plazo de cinco meses, el proyecto se extendió un mes adicional debido a diversas razones, incluyendo pruebas de regresión que consumieron más tiempo del esperado. Aunque no se tomaron medidas específicas para abordar estas dificultades, se reconoce la necesidad de implementar un proceso que garantice estándares de calidad y eficiencia en las pruebas de software. Esta situación evidencia la importancia de mejorar la gestión de pruebas para evitar retrasos y asegurar la satisfacción del cliente. Por lo tanto, el problema radica en la ausencia de un proceso efectivo que permita cumplir con los estándares de calidad y optimizar el proceso de pruebas de software.

Teniendo en cuenta la problemática anteriormente mencionada se establece la siguiente pregunta:

Formulación: ¿Qué impacto puede tener implementar técnicas de automatización del proceso de aseguramiento de calidad en la productividad del desarrollo de software en el marco de los estándares de calidad?

Justificación

La incorporación e integración de la tecnología ha adquirido una mayor relevancia en el marco de la globalización generando una gran necesidad a las empresas por tecnificar, agilizar y transformar digitalmente sus procesos, en busca de mantener y ampliar su competitividad en el mercado de bienes y servicios.

El estudio de las diferentes opciones que se encuentran en el mercado actualmente para realizar pruebas de aseguramiento de calidad de software genera la oportunidad de conocer el potencial de cada una, para proporcionar una solución a la problemática planteada.

Académicamente el proyecto se justifica porque permite aplicar habilidades y destrezas adquiridas en el proceso de formación profesional en Ingeniería Industrial, en busca de contribuir a la mejora continua de procesos con fin de hacerlos más dinámicos, adaptables y competitivos, permitiendo avanzar profesionalmente en las diferentes áreas del conocimiento de la carrera.

Este proyecto como desarrollo personal permite realizar el aporte de las capacidades y apreciaciones al proceso de calidad de software representando un reto de identificar los aportes que se pueden brindar y avances que se lograran.

La comunidad de tecnología se encuentra en cambios constantes solicitando que su equipo se encuentre a la vanguardia para poder brindar a la sociedad que constantemente intenta mejorar su calidad de vida haciendo uso de la tecnología, pero manteniendo su esencia; como empresa se hace necesario que las áreas optimicen tiempos y recursos, este proyecto podrá permitir identificar la ganancia en conocimiento y mejora continua.

De acuerdo con lo anterior, es importante para la empresa considerar los resultados de este proyecto para la toma de decisiones.

Con los resultados obtenidos en el desarrollo de la propuesta, es posible aportar a la empresa algunas alternativas que le permitan:

- Automatizar el proceso de ejecución de pruebas de software a productos, con el fin de optimizar el proceso.
- Determinar los costos asociados al proceso de ejecución de pruebas de software con el fin de ver alternativas para su optimización.
- Establecer métodos para detección y corrección de errores en tiempos oportunos, con el fin de tener opciones de reducir costos asociados a la detección de errores.
- Proponer mejoras en el proceso de aseguramiento de calidad de software.
- Proponer controles para la reducción del riesgo asociados al lanzamiento de un nuevo producto al mercado que no cumpla con el 100% de ejecución del plan de pruebas establecido con el fin de satisfacer las necesidades del cliente.

Objetivos

Objetivo General

Evaluar la eficiencia de la automatización en la productividad del desarrollo de software en el marco de los estándares de calidad.

Objetivos Específicos

Diagnosticar el proceso de pruebas de software que se lleva a cabo en la producción de software en una empresa que desarrolla software.

Implementar pruebas de software automatizadas con el fin de establecer mejoras en la eficiencia del proceso de pruebas de software y su impacto en el ciclo de vida del desarrollo de software.

Evaluar el proceso de automatización de pruebas de software implementado en la organización aplicando estándares de calidad definidos.

Marco Referencial

En esta sección se presentan los temas más importantes que identifican el proyecto a partir de conceptos de pruebas de software hasta la automatización, y en general temas fundamentales utilizados para el desarrollo del proyecto.

Definición de Software

Según Pressman, R. 2010 “El software de computadora es el producto que construyen los programadores profesionales y al que después le dan mantenimiento durante un largo tiempo. Incluye programas que se ejecutan en una computadora de cualquier tamaño y arquitectura, contenido que se presenta a medida que se ejecutan los programas de cómputo e información descriptiva tanto en una copia dura como en formatos virtuales que engloban virtualmente a cualesquiera medios electrónicos.”

Definición de Pruebas de Software

El ISQTB (Internal Software Testing Qualifications Board), una organización sin ánimo de lucro creada en el año 2002 por empresas, instituciones, organizaciones y personas especializadas en el campo de las pruebas y la industria de software define las pruebas como:

“El proceso que consiste en todas las actividades del ciclo de vida, tanto estáticas como dinámicas relacionadas con la planificación, preparación y evaluación de productos de software y productos relacionados con el trabajo para determinar que cumplen los requisitos especificados, para demostrar que son aptos para el propósito y para detectar defectos”.

De acuerdo con Serna, Martínez y Tamayo (2019), en cuanto a las pruebas en software:

> Probar el software es una de las actividades más importantes en el ciclo de vida del desarrollo. La complejidad del software actual exige que la prueba se ejecute de forma

> paralela al desarrollo, de tal manera que los errores se encuentren a tiempo y se puedan corregir a bajo costo La automatización de las pruebas surgió como una alternativa para agilizar su ejecución, a la vez que para mejorar la fiabilidad del producto y su calidad. (p 169).

Para Marin, Trujillo y Buedo (2020) en la industria de desarrollo de software, es necesario asegurar la calidad de los productos y servicios ofrecidos. Para lograr esto, se utilizan modelos, estándares y metodologías ampliamente reconocidos a nivel internacional.

Dentro de los estándares de calidad podemos encontrar:

- ISO/IEC 25000 - SQuaRE (Software product Quality Requirements and Evaluation): En el cual se detallan y describen una serie de normas, con requisitos y métricas que permiten evaluar la calidad de un producto software, incluyendo aspectos como la funcionalidad, fiabilidad, usabilidad, eficiencia, portabilidad y seguridad. (Marcos, Arroyo, Garzás y Piattini, 2008)

- ISO/IEC 9126 - Software engineering - Product quality: Corresponde a un estándar internacional que establece un modelo para la evaluación de la calidad del software basándose en seis características principales: funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad (Al-Kilidar, Cox y Kitchenham, 2005).

- IEEE 829 - Standard for Software Test Documentation: Estándar por el cual se pretende establecer los requisitos y las directrices para la documentación de pruebas de software, incluyendo planes de pruebas, casos de pruebas, informes de pruebas y otros documentos relacionados.

- ISO/IEC 12207 - Software life cycle processes: Este estándar establece los procesos y actividades que se deben llevar a cabo durante el ciclo de vida del software, incluyendo la gestión de requisitos, el diseño, la implementación, las pruebas, la entrega y el mantenimiento.

- ISO/IEC 15504 - Software Process Improvement and Capability Determination (SPICE): Este estándar establece y pretende definir las indicaciones para la implementación de modelo con objetivo en la mejora y los procesos necesarios que permitan la evaluación de software, basado en seis niveles de madurez.

De acuerdo con Pressman, (2005), algunos aspectos a tener en cuenta cuando se valida la calidad del software es el cumplimiento con los requerimientos específicos que fueron definidos de forma clara y explícita en las etapas de diseño y desarrollo, a su vez, identificar las necesidades del usuario que no han sido determinados de manera explícita, pero que pueden ser inferidos por el equipo de desarrollo del software a través de la interacción con el usuario y la comprensión del contexto de uso del software.

Ciclo de Vida de Software

La información es uno de los recursos de mayor relevancia para las empresas, motivo por el cual, la gestión de datos a través de software especializados es cada vez más común y necesaria para el funcionamiento de los procesos productivos de las organizaciones. Este hecho, ha contribuido al crecimiento de la industria de desarrollo de software a nivel mundial, generando un incremento constante en sus ingresos en los últimos años: De acuerdo con cifras de Fernández (2020) durante 2016 este sector tuvo ingresos por $ 70,050 millones de dólares, mientras en 2021 los ingresos aumentaron a $95.385 millones de dólares.

Según la Federación Colombiana de la Industria del Software Fedesoft (2019) en Colombia esta industria presenta un comportamiento similar, pasando de ingresos por $13.466 millones de pesos en 2016 a $16,519 millones de pesos en 2018 y según Asociación Colombiana de Ingenieros de Sistemas (2022), $641 mil millones de pesos en 2021. El auge del desarrollo de software ha generado a su vez, el aumento de la cantidad de empresas dedicadas a esta actividad; según Fedesoft (2019) en 2016 se tenía registro de 6.098 empresas, mientras en 2018 existían 8.569, para 2021 esta entidad registro 9.220 empresas, ubicadas principalmente en Cundinamarca, Antioquia y Valle del Cauca (Centro de investigación especializado en el sector de software y sus relacionados Cenisoft (2021). Este panorama evidencia la necesidad del sector de aumentar la competitividad en el mercado con el fin de generar procesos de desarrollo de software eficaces y eficientes, a través de opciones como la tecnificación y automatización de los procesos de desarrollo a lo largo del ciclo de vida.

El ciclo de vida del desarrollo comprende varias etapas, entre ellas está la del aseguramiento de la calidad donde se verifica y valida que lo que se espera se encuentre funcionando y se pueda incorporar nuevos desarrollos. Día tras día el incremento de desarrollos se va ampliando por lo que se hace necesario realizar un mantenimiento de los módulos de la aplicación ya existentes, garantizando al cliente su compatibilidad con los nuevos requerimientos y que su funcionalidad se encuentra intacta.

En el ciclo de vida de un software se debe tener como foco el cliente – usuario, por lo cual debe ser diseñado y desarrollado para satisfacer las necesidades y expectativas de los usuarios finales y ofrecer una experiencia satisfactoria en cuanto a la usabilidad, sin dejar a un lado los requisitos o especificaciones establecidas en la definición del proyecto, esto se menciona en el artículo IEEE, Standard 610-1990, que hace referencia a la calidad del software.

La norma ISO 12207 define el Ciclo de Vida del Software como Un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso. Existen distintos modelos de Ciclo de Vida, que determinan cuales son y de qué manera se ejecutan las etapas del desarrollo de software. Cada modelo presenta sus propias características y alcances de estas etapas. Para este artículo hemos considerado las siguientes etapas: Expresión de Necesidades y Especificaciones (Requerimientos), Análisis, Diseño, Implementación, Pruebas y Mantenimiento.

El proceso de construcción está formado por etapas que son: la obtención de los requisitos, el diseño del sistema, la codificación y las pruebas del sistema. Desde la perspectiva del producto, se parte de una necesidad, se especifican los requisitos, se obtiene el diseño de este, el código respectivo y por último el sistema de software. Algunos autores sostienen que el nombre ciclo de vida ha sido relegado en los últimos años, utilizando en su lugar proceso de software, cambiando la perspectiva de producto a proceso. [J. Juzgado, 1996]

El software o producto, en su desarrollo pasa por una serie de etapas que se denominan ciclo de vida, siendo necesario, definir en todas las etapas del ciclo de vida del producto, los procesos, las actividades y las tareas a desarrollar.

Por lo tanto, se puede decir que se denomina ciclo de vida a toda la vida del software, comenzando con su concepción y finalizando en el momento de la desinstalación de este. [Sigwart y col., 1990], aunque a veces, se habla de ciclo de desarrollo, para denominar al subconjunto del ciclo de vida que empieza en el análisis y finaliza la entrega del producto.

Un ciclo de vida establece el orden de las etapas del proceso de software y los criterios para tener en cuenta para poder pasar de una etapa a la siguiente.

Clasificación de Pruebas de Software

Para certificar la calidad de software que desarrolla la empresa dentro del ciclo de vida, se cuenta con una etapa de pruebas, las cuales actualmente se usa el tipo de pruebas funcionales, lo cual permite entregar un producto al cliente certificado, sin embargo, el software ha crecido y los tiempos en pruebas aumentan, estos podrían ser optimizados al aplicar la automatización de pruebas de software.

Pruebas funcionales: “Se basan en funciones, prestaciones y en su interoperabilidad con sistemas específicos, y pueden llevarse a cabo en todos los niveles de prueba” (Mera, 2016, p.170). Se orientan en la evaluación mediante técnicas de tipo caja negra.

Técnicas de Pruebas de Software

Dentro del proceso de pruebas automatizadas, se encuentran dos agrupaciones principales de acuerdo con Mera (2016), las cuales son la técnica de caja blanca y técnica de caja negra.

Para la problemática identificada en la empresa, y para el caso de estudio la técnica usada actualmente se enfoca en la caja negra.

Técnica de Caja Negra:

La técnica de caja negra (también conocida como prueba funcional o de comportamiento) permite realizar pruebas de software en la que se evalúa el funcionamiento del software sin conocer su estructura interna o su lógica de programación. Es decir, se trata de una prueba basada únicamente en la entrada y salida del software, sin considerar cómo se realiza el procesamiento de datos dentro del programa.

Para llevar a cabo la técnica de caja negra, se pueden utilizar diferentes técnicas de diseño de pruebas, como la partición de equivalencia, la prueba de valores límite, la prueba de casos de uso, entre otras.

Los niveles en el proceso de pruebas funcionales, para el desarrollo de software, es posible identificar:

Prueba de componentes o unitarias: “Tienen por objeto localizar defectos y comprobar el funcionamiento de módulos software, programas, objetos, clases, etc, que puedan probarse por separado. Es decir, se pueden realizar de manera independiente al resto del sistema en función del contexto.” (Mera, 2016, p.169).

Pruebas de integración: “Se encargan de probar las interfaces entre los componentes o módulos; por ejemplo, el componente validación de usuario con el sistema operativo, el sistema de archivos en integración con el hardware” (Mera, 2016, p.169).

Pruebas de aceptación: “Se enfocan en la aceptación de los criterios previstos en un contrato de desarrollo de software, acordado entre la fábrica de software y el cliente” (Mera, 2016, p.169).

¿Qué es Automatización de Pruebas de Software?

La automatización de pruebas de software es una técnica que consiste en utilizar herramientas de software especializadas para llevar a cabo pruebas de software de forma automatizada. Esto se logra mediante la creación de scripts o programas que simulan la interacción de un usuario con una aplicación o sistema, y que son capaces de evaluar el comportamiento del sistema ante diferentes entradas y condiciones. La automatización de pruebas es una práctica cada vez más extendida en el ámbito del desarrollo de software, ya que

permite reducir los costos y el tiempo necesario para llevar a cabo pruebas exhaustivas y repetitivas.

Como lo menciona Rehkopf en su trabajo del 2023, destaca el uso de herramientas automáticas para automatizar el proceso manual de revisión y validación del software desarrollado, lo que implica definir una herramienta especifica de automatización que no solo ejecute las pruebas, sino que también genere un reporte con los hallazgos identificados. Esto en aras de mejorar y optimizar el proceso de validación del software, que haciéndose manualmente puede ser susceptible a errores y representa un alto rubro para las empresas, y mejorando los tiempos de entrega de los desarrollos a los clientes.

Para Colorado (2020):

La automatización y digitalización de los procesos en los negocios, constituyen una de las áreas fundamentales de la transformación digital, las empresas dedicadas a esta labor tienen el reto de suplir las necesidades que hoy existen en cuanto a minimizar los tiempos de espera en las entregas de software de calidad y el aprovechamiento al máximo de las nuevas tecnologías (p 4).

Metodología Lean Manufacturing

La filosofía Lean Manufacturing, también conocida como Lean Production, es un sistema de organización del trabajo que pone el foco en la mejora del sistema de producción. Para esto se basa en la eliminación de aquellas actividades que no aportan valor al proceso ni al cliente. Estas se denominan despilfarros o desperdicios, y son aquellas tareas que implican la sobreproducción, altos tiempos de espera o desperfectos en los productos, por citar algunos ejemplos. (Andreu, 2023)

La filosofía Lean Manufacturing trata de optimizar el sistema de producción y reducir o eliminar las tareas que no añadan valor. (Andreu, 2023)

El término Lean Manufacturing apareció por primera vez en los años 70 en el libro La máquina que cambió el mundo de los autores Womack, Jones y Ross. La obra, que se ha convertido en best seller mundial, fue la primera que sacó a la luz el sistema de producción lean de Toyota. Este contrapone dos sistemas de negocio radicalmente diferentes: producción lean versus producción en masa. (Andreu, 2023).

Metodología Seis Sigma

Six Sigma es una metodología de mejora de procesos que ayuda a las organizaciones a perfeccionar sus procesos de negocios. Six Sigma se aplica, fundamentalmente, para establecer la uniformidad en los procesos a fin de reducir la cantidad de variaciones del producto final. En definitiva, con este método se minimizan los defectos de un producto. (Laoyan, 2024).

La principal filosofía de Six Sigma indica que todos los procesos se pueden definir, medir, analizar, mejorar y controlar (lo que comúnmente se conoce como el método DAMAIC, por sus siglas en inglés). Según Six Sigma, en todos los procesos debe haber entradas y salidas.

Las entradas son acciones que el equipo lleva a cabo y las salidas son los efectos de esas acciones. La idea central es que si puedes controlar la mayor cantidad de entradas (o acciones) como sea posible, también controlarás sus salidas. (Laoyan, 2024).

Componentes

Mejora Continua (KAIZEN): El término "Kaizen" fue acuñado por Massaki Imai en 1986 y desde entonces su uso se ha estandarizado y popularizado, al menos geográficamente, aunque no todas las empresas lo han adoptado en la misma medida. Este término está relacionado con la mejora continua, la eliminación de desperdicios, la estandarización de procesos y otros pilares del Lean Manufacturing. Así pues, los programas de mejora continua

tienen como objetivos principales: Aumentar el nivel de calidad, Mejorar la satisfacción del cliente (con disminución de las No Conformidades de clientes), Optimización de la gestión de la empresa e Incrementar en el rendimiento de equipos humanos. (Imai, 1986)

Just Intime: En la metodología Lean clásica, un sistema Justo a Tiempo implica el uso de una estrategia de flujo de línea para alcanzar una producción de gran volumen a un bajo costo. El objetivo es mantener un proceso de producción continuo, sin interrupciones, con el fin de minimizar el tiempo total que transcurre desde el inicio de la fabricación hasta la entrega del producto final y su facturación. Los 7 pilares de justo a tiempo son los siguientes: Igualar la oferta y la demanda, El peor enemigo: el desperdicio, El proceso debe ser continuo no por lotes, Mejora Continua, Es primero el ser humano, La sobreproducción = ineficiencia y No vender el futuro. (Hibbs, 2009).

Visual Streaming Map (VSM): El propósito del mapeo de la cadena de valor es para hacer resaltar la causa del desperdicio y eliminarlos para la implementación de un estado futuro de la cadena de valor que puede convertirse en realidad en un periodo corto de tiempo. El mapa de la cadena de valor representa en un diagrama el conjunto de actividades y procesos y el flujo de materiales e información que rodean el proceso transformación de un producto o prestación de un servicio desde que se recibe la petición de este hasta que se realiza su entrega, pudiendo incluir actividades de clientes y proveedores que intervengan en el proceso de producción (Sanz, 2015).

Gestión Visual: La gestión visual es una herramienta complementaria a la labor de las personas, ya que aprovecha la naturaleza sensible de los seres humanos para transmitir información de manera efectiva. Los indicadores visuales son los más eficaces para mostrar el estado de cumplimiento de las tareas y procesos, ya que se valen de los sentidos humanos, como

el oído y la vista, para presentar información de manera clara y comprensible. En la actualidad, la gestión visual se está aplicando cada vez más en el ámbito del desarrollo de software. Dado que, consiste en presentar información relevante de manera gráfica y sintética para facilitar el trabajo de los miembros del equipo y reducir tiempos. Al utilizar controles visuales eficaces, se mejora la productividad, se reducen los errores, se fomenta la comunicación y se aumenta el control del entorno laboral. En este aspecto, la filosofía "Lean" enfatiza la importancia de los controles visuales para prevenir que los problemas pasen desapercibidos (Dingsoyr y Moe, 2014).

Jidoka: La filosofía Jidoka persigue la determinación de parámetros de calidad óptimos en los procesos de producción y se enfoca en integrar la verificación de calidad en el proceso. A través del sistema Jidoka, se comparan los parámetros del proceso con los estándares preestablecidos y, si no coinciden, el proceso se pone en pausa con una alerta de irregularidad, entonces se debe solucionar el problema para evitar la producción de productos defectuosos.

Estos procesos son comparativos entre los resultados actuales en producción y lo "ideal" o "estándar". Por otro lado, la implantación de diferentes tipos de sistemas Jidoka depende del tipo de producto y la información "ideal" o "estándar" debe ser el punto óptimo de calidad del producto. Igualmente, Jidoka también puede referirse a equipos que se detienen automáticamente en condiciones anormales o cuando un colaborador encuentra un problema en su estación de trabajo (Caudillo, 2013).

Heijunka: Conocida en español como Producción Nivelada, según Hernández (2011), se refiere a la técnica de nivelar la producción, tanto en términos de volumen como de mezcla de productos. En lugar de fabricar según el flujo de pedidos de los clientes, que puede ser variable, se considera el volumen total de pedidos en un período determinado y se equilibra la producción de manera que se fabrique la misma cantidad de cada tipo de producto todos los días para cubrir

la demanda global. La metodología ágil incorpora el concepto de "Heijunka", que se enfoca en la versatilidad de las personas para desempeñar diversas tareas. Cuanto mayor sea la capacidad de los miembros del equipo para realizar diferentes tareas, más fácil será nivelar la producción de software (Sanz, 2015).

Personas y Equipo de Trabajo: Según Poppendieck (2003), el Lean Software Development (LSD) se enfoca en la colaboración y la comunicación efectiva entre el equipo de trabajo, lo que se traduce en una mejora en la calidad del software y en una reducción del tiempo de entrega. Siguiendo esta línea, Liker (2021) menciona que las principales características comunes a los equipos de trabajo Lean incluyen grupos reducidos por tareas, un responsable o líder Lean por cada equipo en el que recaen responsabilidades grupales, el uso de controles visuales de manera estricta, el enfoque en tareas específicas o temas concretos, y la realización de actividades y reuniones planeadas y metódicas para garantizar controles y mejora continua.

3.9 Benchmarking

Proceso sistemático y continuo para evaluar los productos, servicios y procesos de trabajo de las organizaciones que son reconocidas como representantes de las mejores prácticas, con el propósito de realizar mejoras organizacionales (Tijerina,1999).

ISO/IEC 25010

Norma internacional que define un modelo de calidad para productos de software. Este modelo se utiliza para evaluar y mejorar la calidad del software, proporcionando un marco estructurado que abarca diversas características y sus características de calidad. Entre las

principales características se incluyen la funcionalidad, la fiabilidad, la usabilidad, la eficiencia, la mantenibilidad, la portabilidad, la compatibilidad y la seguridad. Cada una de estas características se descompone en subcaracterísticas específicas que permiten una evaluación detallada y precisa. La función de la ISO/IEC 25010 es proporcionar una guía estandarizada que ayuda a los desarrolladores y evaluadores de software a identificar, medir y mejorar los aspectos críticos de la calidad del software, asegurando que el producto final cumpla con las expectativas y requisitos de los usuarios y otras partes interesadas.

Diseño Metodológico

En este proyecto se trabajará sobre la metodología Lean Manufacturing de acuerdo con la investigación realizada esta es la más acorde para la temática que se va a desarrollar.

Para esta selección se hizo un cuadro comparativo entre las dos metodologías:

Tabla 1

Comparación de Metodologías

Factor	Lean Manufacturing	Seis Sigma
Alcance	Eliminar el desperdicio y aumentar el flujo de valor.	Reducir la variabilidad y aumentar la calidad.
Enfoque	Eliminar actividades que no agregan valor.	Reducir defectos y errores.
Metodología	Planificar-Hacer-VerificarActuar (PHVA).	Definir-Medir-Analizar-Mejorar-Controlar (DMAIC).
Herramientas	Mapeo del flujo de valor (VSM), análisis de las 5S, Kanban, Jidoka.	Diagramas de Pareto, análisis de capacidad de procesos, diseño de experimentos.
Ventajas	Reduce costos, aumenta la eficiencia, mejora la productividad.	Reduce defectos, aumenta la calidad, mejora la satisfacción del cliente.
Desventajas	Puede ser difícil de implementar, requiere un cambio cultural.	Puede ser costoso y complejo de implementar.
Métricas	Tiempo de ciclo, Lead time, Eficiencia, Calidad, Flexibilidad.	Defectos por millón de oportunidades (DPMO), Capacidad del proceso (Cpk), Tiempo de ciclo del proceso, Variación.

Nota. Comparación de metodologías. *Fuente*. Basado en las metodologías Lean Manufacturing y Seis Sigma.

La metodología Lean manufacturing se ha implementado en diferentes tipos de empresas a pesar de que su origen se dio en la manufactura, esto con el fin de alcanzar resultados cuantificables y satisfactorios. Los objetivos de la metodología Lean son:

- Limitar el desperdicio
- Priorizar a las personas por encima de los procesos

- Hacer entregas rápidas y frecuentes
- Emplear la perspectiva del cliente en el diseño del producto
- Introducir mejoras en todas las áreas, no sólo en algunas
- Aprovechar todas las oportunidades para mejorar que se presenten.

El pensamiento de proceso Lean se centra en las necesidades del cliente. El cliente identifica un valor en su proceso, mapea el flujo de valor del proceso en función de ese valor y crea un flujo basado en extracción. A lo largo del ciclo de vida del proyecto, se monitorean y abordan las oportunidades de mejora para garantizar el crecimiento continuo y la optimización de los procesos.

Filosofía

1. Basar las decisiones de administración en una filosofía de largo plazo, aún a costo de las metas financieras de corto plazo. Procesos.
2. Crear flujos de procesos continuos para llevar los problemas a la superficie.
3. Usar sistemas "Pull" para evitar la sobreproducción.
4. Nivelar la carga de trabajo.
5. Construir una cultura orientada a la solución de problemas, para obtener calidad a la primera vez.
6. La estandarización de tareas y procesos es la base de la mejora continua y la toma de poder por los empleados.
7. El control visual impide que se oculten los problemas.
8. Utilizar tecnología fiable y testeada que sea de utilidad para las personas y los procesos. Personas y Colaboradores

9. Desarrollar líderes que entiendan su trabajo en Toyota, vivan su filosofía y se la enseñen al resto.

10. Desarrollar personas y equipos excepcionales que sigan la filosofía de la compañía.

11. Se debe respetar la red de colaboradores y proveedores, dándoles nuevos retos y ayudándolos a mejorar. Resolución de Problemas

12. Para comprender una situación se debe verificar en primera persona.

13. Tomar decisiones lentas por consenso, considerar profundamente todas las opciones e implementar las decisiones rápidamente.

14. Por medio de la reflexión implacable (Hansei) y la mejora continua (Kaizen) la empresa debe asumir un rol de aprendizaje sistemático.

Metodología de Lean Software Development

Lean Software Development es una adaptación de los principios y prácticas del Lean Manufacturing al dominio del desarrollo de software. Al igual que en la fabricación, el objetivo es minimizar el desperdicio y maximizar el valor para el cliente.

Pasos

1. Eliminar desperdicios: Hacer desaparecer del proceso y el producto todo aquello que no aporta valor al cliente. (Rubio, 2023)

2. Asegurar la calidad: Se enfoca en la calidad desde el inicio del proceso y se implementan acciones correctivas preventivas para evitar errores. (Rubio, 2023)

3. Crear conocimiento: El proceso de desarrollo de software es una creación constante de conocimiento, y se debe centrar en mejorarlo y profundizarlo. (Rubio, 2023)

4. Aplaza el compromiso: Se recomienda retrasar la toma de decisiones debido a la incertidumbre que rodea los requisitos. (Rubio, 2023)

5. Entrega rápida: Se busca entregar el software tan rápido como sea posible sin sacrificar la calidad. (Rubio, 2023)

6. Respeto a la gente: Se debe reconocer y respetar a todas las personas involucradas en el proceso de desarrollo. (Rubio, 2023)

7. Optimiza la totalidad: Se evita la mejora local en favor de una visión global del proceso de desarrollo del software. (Rubio, 2023).

Tabla 2

Relación objetivos con pasos de la metodología

Paso de la metodología	Relación con objetivos	Entregables
Eliminar el Desperdicio (Waste):	● Diagnosticar el proceso de pruebas de software que se lleva a cabo en la producción de software en una empresa que desarrolla software. ● Implementar pruebas de software automatizadas con el fin de establecer mejoras en la eficiencia del proceso de pruebas de software y su	- Realizar VSM (Mapa de flujo de valor) - Implementar la automatización caso de uso.

	impacto en el ciclo de vida del desarrollo de software.	
Ampliar el Aprendizaje (Amplify Learning):	● Diagnosticar el proceso de pruebas de software que se lleva a cabo en la producción de software en una empresa que desarrolla software. ● Implementar pruebas de software automatizadas con el fin de establecer mejoras en la eficiencia del proceso de pruebas de software y su impacto en el ciclo de vida del desarrollo de software.	Flujo de proceso hoy y el realizado en el proyecto.
Decidir lo más tarde posible (Decide as Late as Possible):	● Implementar pruebas de software automatizadas con el fin de establecer mejoras en la eficiencia del proceso de pruebas de software y su impacto en el ciclo de vida del desarrollo de software.	Análisis de resultados

Entregar lo más rápido posible (Deliver as Fast as Possible):	● Implementar pruebas de software automatizadas con el fin de establecer mejoras en la eficiencia del proceso de pruebas de software y su impacto en el ciclo de vida del desarrollo de software.	Análisis de resultados
Empoderar al Equipo (Empower the Team):	● Diagnosticar el proceso de pruebas de software que se lleva a cabo en la producción de software en una empresa que desarrolla software. ● Implementar pruebas de software automatizadas con el fin de establecer mejoras en la eficiencia del proceso de pruebas de software y su impacto en el ciclo de vida del desarrollo de software.	Flujo de procesos. Análisis de resultados.
Construir Integridad (Build Integrity In) Optimizar el Todo (Optimize the Whole)	● Evaluar la eficiencia de la automatización en la productividad del desarrollo de software en el marco de los estándares de calidad.	Análisis de resultados y recomendaciones.

Nota. Pasos de la metodología relacionada con los objetivos. *Fuente*. Rubio (2023)

Desarrollo de Objetivos

Diagnóstico del Proceso de Pruebas de Software

Contextualización de la Empresa

El siguiente organigrama representa como se encuentra la empresa organizada de acuerdo con sus áreas y conexiones entre las mismas.

Figura 1

Organigrama de la empresa

Nota. Organigrama empresarial. *Fuente*. Tomado de la empresa que desarrolla Software (2024).

En el siguiente organigrama representa como se encuentra el área de tecnología organizada de acuerdo con sus áreas y conexiones entre las mismas.

Figura 2

Organigrama de tecnología

Nota. Organigrama del área de Tecnología. *Fuente.* Basado en datos de la empresa que desarrolla Software (2024).

En la siguiente figura se observará los procesos misionales de la empresa que desarrolla software.

Figura 3

Procesos misionales de la empresa

Nota. Figura de procesos misionales de la empresa. *Fuente*. Basada en información dada por la empresa que desarrolla software (2024).

Diagnóstico del Proceso de Pruebas de software

Se ha elaborado la siguiente tabla de convenciones que detalla los íconos utilizados y su significado en los flujogramas realizados representando el estado actual del diagnóstico del proceso de pruebas de software.

Tabla 3

Convecciones íconos de flujogramas

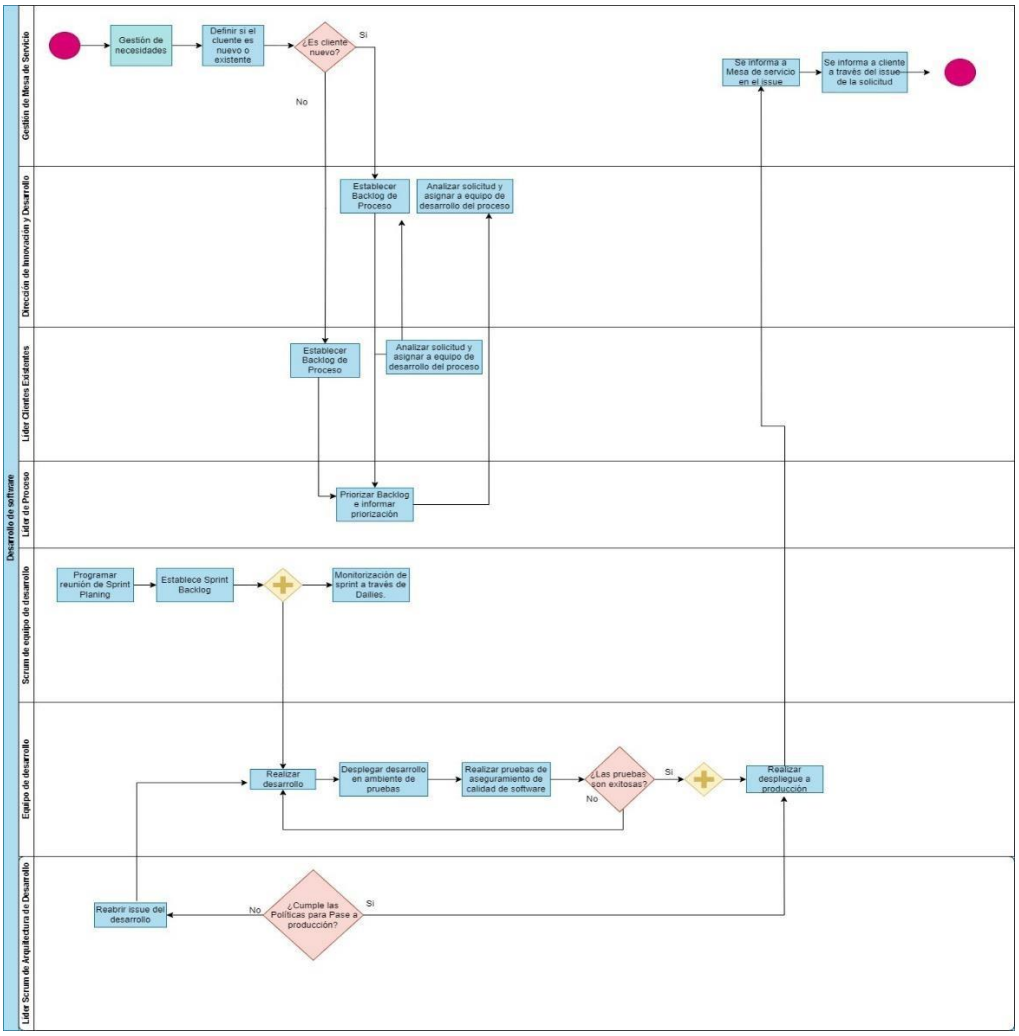
Ícono	Nombre	Descripción
	Evento de Inicio	Indica el punto de inicio del proceso.
	Evento Intermedio	Representa algo que ocurre entre el inicio y el fin del proceso, como una interrupción o un mensaje recibido.
	Evento de Fin	Marca el final del proceso.
	Actividad	Una tarea o trabajo realizado dentro del proceso.
	Subproceso	Un conjunto de tareas agrupadas que pueden ser desglosadas en un diagrama separado.
	Gateway	Decisión o punto de bifurcación en el flujo del proceso.
	Conector	Línea que conecta diferentes elementos del diagrama, mostrando el flujo de secuencia.
	Piscina (Pool)	Representa a un participante o entidad en el proceso, que puede contener varios carriles (Lanes).
	Carril (Lane)[i]	Subdivisión dentro de una piscina que representa diferentes roles o departamentos.
	Evento de Terminación	Indica que un proceso o subproceso ha terminado, y no se ejecutará ninguna otra actividad.

Nota. Se realiza la convección de iconos usados en flujograma. *Fuente*. Basado en Símbolos y notación de diagramas BPMN.

En el siguiente flujograma se mostrará el proceso sé que realiza en el equipo de tecnología desde que se inicia recibe el requerimiento del cliente hasta que se publica en producción para el cliente.

Figura 4

Flujograma de tecnología

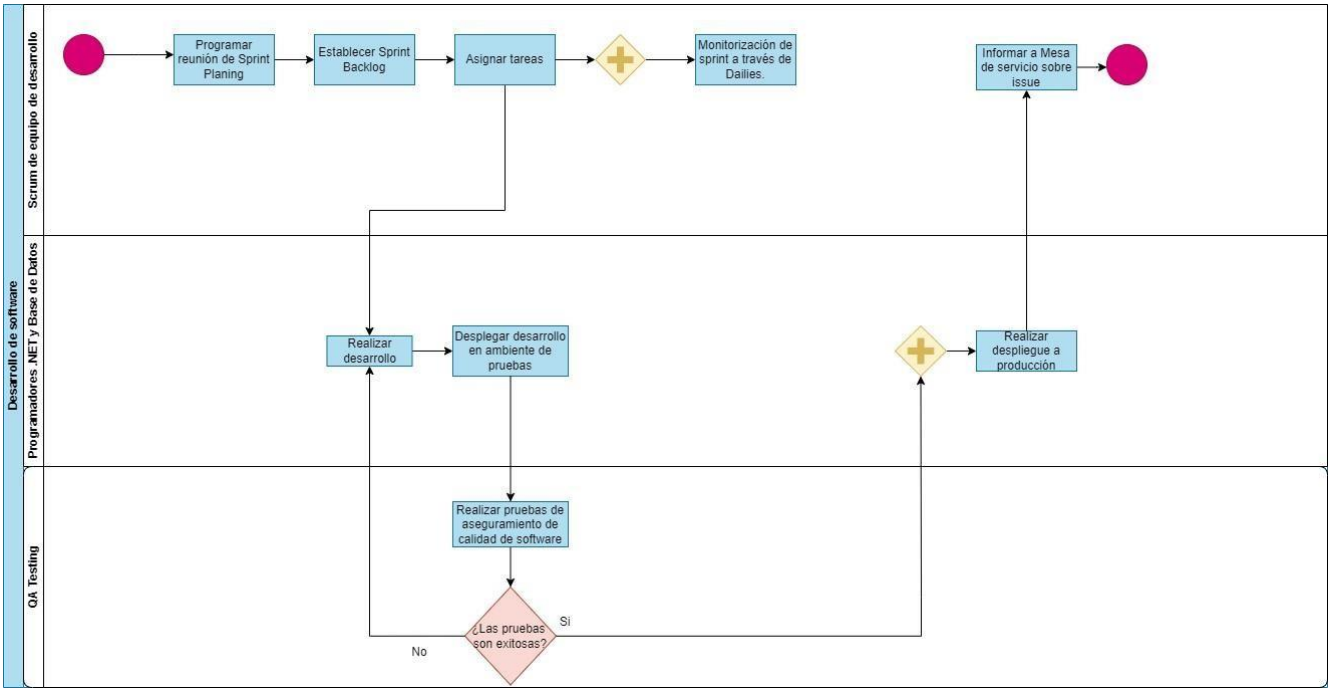


Nota. Flujograma del proceso que realiza Tecnología. *Fuente.* Basado en datos de la empresa que desarrolla software (2024)

En el siguiente flujograma se mostrará el proceso sé que realiza en la célula desde que se inicia a trabajar en el requerimiento del cliente hasta que se publica en producción para el cliente.

Figura 5

Flujograma de la célula

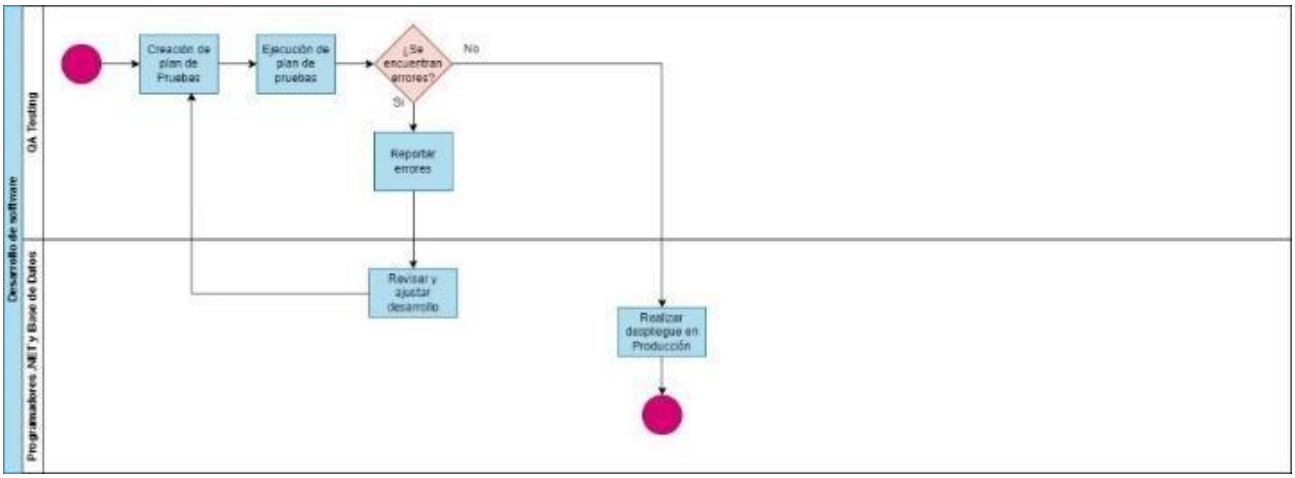


Nota. Flujograma del proceso que realiza un equipo (célula) de Tecnología. *Fuente.* Basado en datos de la empresa que desarrolla software (2024).

En el siguiente flujograma se mostrará el proceso sé que realiza en Pruebas de software desde la funcionalidad que se tiene para pruebas de calidad hasta que se publica en producción para el cliente.

Figura 6

Flujograma de pruebas de calidad de software

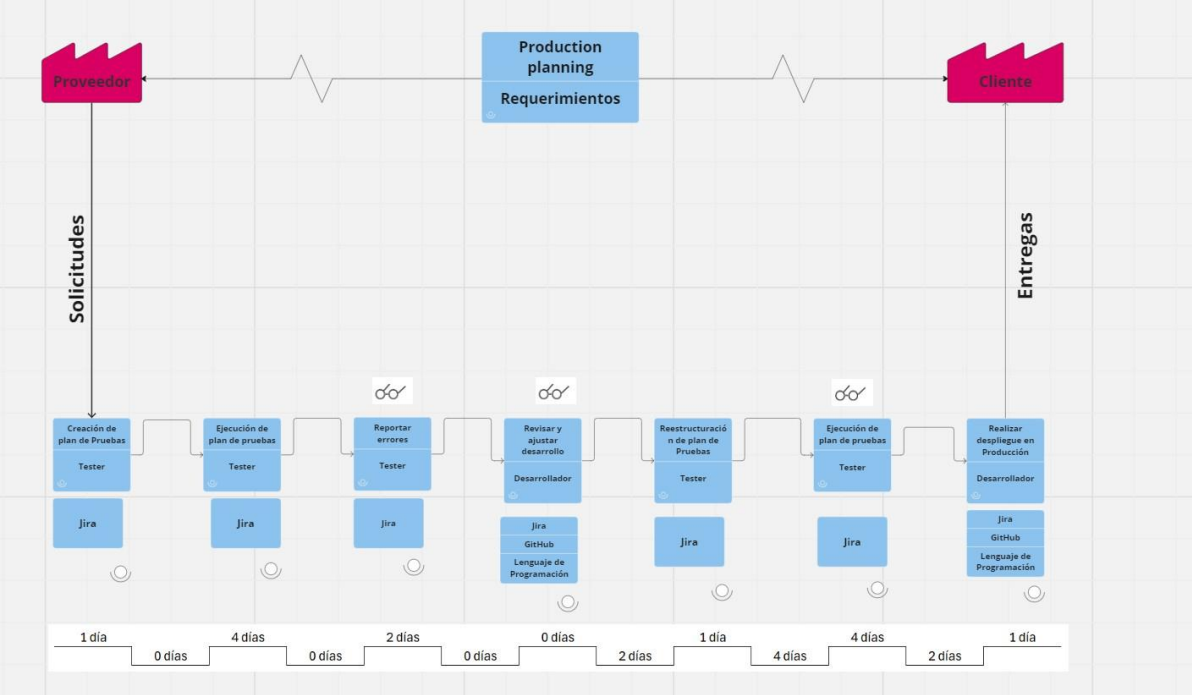


Nota. Flujograma del proceso de Pruebas de calidad de software. *Fuente.* Basado en datos de la empresa e ISQTB (2024)

En el caso del proceso de pruebas de software, su VSM (Mapa de flujo de valor) inicia con el requerimiento para pruebas donde el Tester realiza la creación del plan de pruebas en 1 días. A partir de allí, inicia la ejecución de pruebas; se procesa la ejecución se realiza aproximadamente en 4 días; al encontrar fallas se reportan en 2 días y pasan 2 días de inactividad; se reestructura el plan de pruebas y pasan 4 días de inactividad; se ejecuta de nuevo en 4 días, pasan 2 días de inactividad y, finalmente, llega al cliente.

Figura 7

Mapa de flujo de valor



Nota. Mapa de valor que realiza un equipo (célula) de Tecnología. *Fuente*. Basado en datos de la empresa que desarrolla software (2024)

Análisis del VSM:

En el diagrama de proceso de pruebas de software se ha detectado que en la ejecución de pruebas se está trabajando y es la etapa que más tiempo requiere, por eso agregó los lentes de «Revisar». Los Tester no dan abasto para tener listos los planes de pruebas, ejecutarlos, reportar y entregar los requerimientos. La conclusión es que se requiere una mejora en el proceso de ejecución de pruebas.

Desperdicios identificados:

- Tiempos muertos.
- Códigos abandonados.
- Desarrollos que no se publican.
- Reprocesos.
- Esperas (posible retroalimentación).
- Defectos de calidad.

Toma de Tiempos en Ejecución de Pruebas Manuales de Software

La tabla presentada a continuación constituye el resumen de la ejecución del plan de pruebas de una unidad de negocio, entendiendo por unidad de negocio a un módulo que hace parte de un proyecto de software más amplio. Esto módulos en algunos casos son trasversales, es decir que son usados por diferentes objetos de negocio, por ejemplo, la unidad de negocio de facturación.

El plan objeto de la muestra está compuesto por 15 pasos diseñados (Ver anexo 1), se llevó a cabo para evaluar con precisión la funcionalidad. Este módulo es fundamental para la realización de diversas acciones, dentro de la aplicación.

Tabla 4

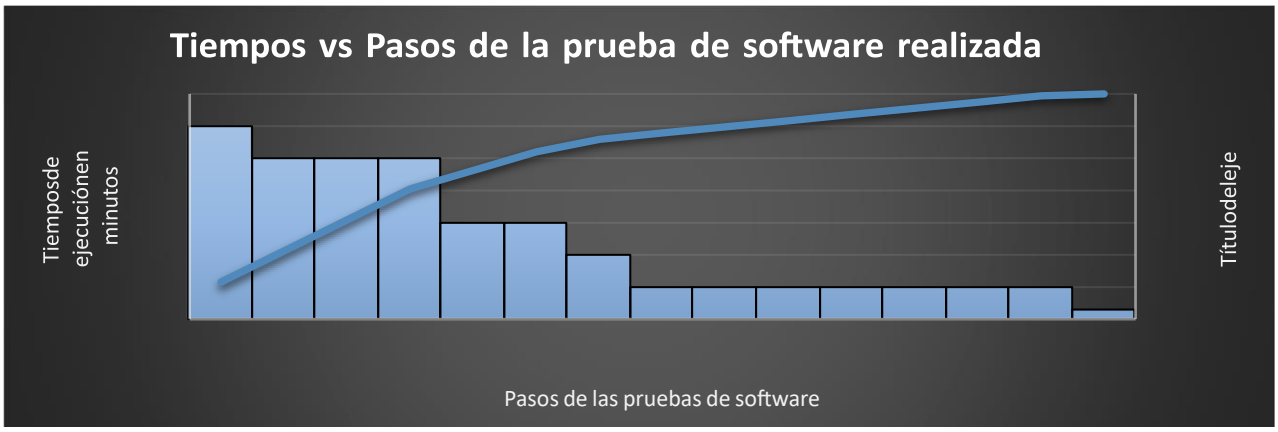
Resumen de Ejecución del Plan de Pruebas

Nombre del Test	Duración (min)	Observaciones
Paso 1	0,6	Retraso por ambiente de pruebas
Paso 8	0,5	Retraso en revisión de data y ambiente
Paso 9	0,5	Retraso en revisión de data y ambiente de pruebas
Paso 11	0,5	Retraso en revisión de data y ambiente de pruebas
Paso 7	0,3	Retraso en revisión de data
Paso 12	0,3	Retraso en ambiente de pruebas
Paso 2	0,2	Sin novedad
Paso 3	0,1	Sin novedad
Paso 4	0,1	Sin novedad
Paso 5	0,1	Sin novedad
Paso 6	0,1	Sin novedad
Paso 13	0,1	Si novedad
Paso 14	0,1	Sin novedad
Paso 15	0,1	Sin novedad
Paso 10	0,03	Sin novedad
Total	3, 63	

Nota. Se realiza la relación de los tiempos tomados en cada paso realizado en la prueba ejecutada manualmente. *Fuente*. Autoría Propia

Gráfica 1

Tiempos vs Pasos de la prueba de software realizada.



Nota. Este gráfico representa la relación de los tiempos empleados en cada paso de la prueba manual realizada. *Fuente*. Autoría Propia.

El gráfico de "Tiempos vs Pasos de la prueba de software", los pasos P.1, P.8, P.9 y P.11 destacan por sus tiempos de ejecución más altos, indicando áreas que requieren atención para optimizar la eficiencia. La mayoría de los otros pasos tienen tiempos de ejecución significativamente menores, con P.10 como el más rápido. Este análisis sugiere que, para optimizar la prueba, se debe enfocar en reducir los tiempos de los pasos más largos.

Según la tabla número 8, de la toma de tiempos de los pasos realizados en el test de pruebas, se puede concluir lo siguiente:

El tiempo promedio de ejecución de una prueba manual es de aproximadamente 3,6 minutos por cada test. Siendo el máximo 0,6 minutos y el mínimo 0,03 minutos.

El paso 1 fue el más demorado porque el ambiente presento latencia y al consultar la data de la factura genero demoras.

En condiciones normales un set de pruebas incluye en promedio de 20 test (un flujo completo) en un plan de pruebas para verificar el correcto funcionamiento de un componente. Este flujo puede ser el inicio de varios flujos de la aplicación. De acuerdo con esto, el tiempo total estimado es:

20 test × 3.6 minutos/escenario = 72 minutos

Dependiendo el módulo del software que se esté probando, la prueba completa puede contener uno o más flujos, entendiendo por flujo un set completo como se muestra en la tabla número 8. En promedio en el software desarrollado en la empresa objeto de este análisis está compuesto de 8 flujos. De tal manera que:

8 flujos x 72 minutos (set de pruebas) = 576 minutos equivalente a 9,6 horas

En horas ingeniero sería:

9,6 x $30.500[1]* = $292.800

Realizar Pruebas Automatizadas de Software.

Identificación de la Herramienta para Automatizar Pruebas de Software

Evaluación de Herramientas de Automatización de Pruebas de Software a través de Benchmarking.

[1] * Valor corresponde al asignado en el momento de realizar el documento en la empresa de desarrollo de software analizada.

El benchmarking funcional comprende la identificación de productos, servicios y procesos de trabajo de organizaciones. El objetivo del benchmarking funcional es identificar las mejores herramientas de automatización de pruebas de software (Tijerina, 1999, p.16).

Las empresas emplean el benchmarking con diversos objetivos. Para algunas organizaciones, constituye una parte esencial de un proceso integral de resolución de problemas, con el propósito de mejorar la organización en su totalidad. Otras lo ven como un mecanismo activo para mantenerse al día con las prácticas modernas del negocio. En el contexto de esta investigación, se utilizará el benchmarking para determinar qué herramienta de software de testing se ajusta mejor a las necesidades de la empresa de desarrollo de software.

Algunas de las razones por las cuales las organizaciones emplean el proceso de benchmarking son (Tijerina, 1999, p.17):

- Planeación estratégica.
- Pronósticos.
- Comparaciones de Producto/Proceso
- Fijación de Objetivos.

Antes, el benchmarking solía centrarse en temas relacionados con productos. No obstante, a medida que ha crecido la experiencia con esta práctica, se han ampliado considerablemente las áreas potenciales de investigación.

Las categorías de información que se presentan a continuación no son una lista exhaustiva de las áreas que pueden ser sometidas a benchmarking, pero sí abarcan las áreas más comunes a las que se recurre en busca de información (Tijerina, 1999, p.20):

- Productos y servicios.

- Procesos de trabajo.
- Funciones de apoyo.
- Desempeño Organizacional.
- Estrategia

Etapas

1. Determinar aplicaciones de software a evaluar: Es identificar las herramientas de automatización de pruebas de software a las cuales se les va a hacer el benchmarking.

2. Seleccionar las fuentes de información: Consiste en identificar fuentes de información acerca de las herramientas de automatización de pruebas de software que se tendrán en cuenta. Y obtener datos detallados sobre cada una de las herramientas seleccionadas. Esto puede incluir características, funcionalidades, precios, opiniones de usuarios y cualquier otra información relevante disponible públicamente.

3. Resumir los datos obtenido en el benchmarking: Se destacan los datos más relevantes y significativos que se han recopilado durante el benchmarking. Esto puede incluir resultados sobresalientes, tendencias importantes, diferencias significativas entre las entidades comparadas y otras observaciones destacadas.

4. Seleccionar herramienta de automatización de pruebas de software: Después de los anteriores pasos, se selecciona la herramienta más eficiente para el proceso de pruebas de software.

Tabla 5

Etapas de Benchmarking

Etapa	Actividades
Determinar aplicaciones de software a evaluar.	- Establecer los criterios de evaluación de cada herramienta de automatización de pruebas de software. - Identificar herramientas de automatización de pruebas de software. - Seleccionar herramientas de automatización.
Seleccionar las fuentes de información.	- Consultar las páginas de documentación de las herramientas de automatización de pruebas de software. - Determinar las fuentes mas relevantes de acuerdo con los criterios a tener cuenta. - Evaluar cualitativa y cuantitativamente cada herramienta con la información recopilada.
Resumir los datos obtenido en el benchmarking.	- Determinar los datos más relevantes. Sistematizar los datos relevantes obtenidos.
Seleccionar la herramienta de automatización de pruebas de software.	- Definir la herramienta de automatización de pruebas de software de acuerdo con el análisis obtenido.

Nota. Se describen las etapas que se desarrollaran en la investigación Benchmarking. *Fuente*.

Basado en BENCHMARKING - METODOLOGÍA DE DESARROLLO Y APLICACION

(1999)

Tabla 6

Formato de Evaluación de Herramientas

Herramientas para Automatizar Pruebas de Software				
Aplicación	**Parámetro de Evaluación**	**Valor Cualitativo**	**Escala Cuantitativa**	**Interpretación**
Nombre de la herramienta	Acceso a documentación técnica	Total | Parcial | Nula	0-10	Total (8-10): Acceso a la documentación técnica es completo y exhaustivo. Los usuarios pueden encontrar la información necesaria de manera fácil y rápida. Parcial (4-7)): El acceso a la documentación técnica es limitado. Los usuarios pueden encontrar parte de la información que necesitan, pero faltan detalles importantes o la información esté desactualizada. Nula (0-3): No hay acceso a la documentación técnica en absoluto. Los usuarios no pueden encontrar ninguna información útil que les ayude a utilizar la herramienta o resolver problemas.
	Soporte de la herramienta (empresa-comunidad)	Bueno | Regular | Nulo	0-10	Bueno (8-10): El soporte de la herramienta es excelente, tanto por parte de la empresa como de la comunidad. Los usuarios reciben respuestas rápidas y útiles a sus consultas, y hay una sólida colaboración entre la empresa y la comunidad. Regular (4-7): El soporte de la herramienta es aceptable. Las respuestas son lentas o menos detalladas, y la colaboración entre la empresa y la comunidad es mínima. Nulo (0-3): El soporte de la herramienta es inexistente. Los usuarios no reciben ninguna ayuda o respuesta a sus consultas, ya sea por parte de la empresa o de la comunidad.
	Licencias	Total | Parcial | Nula	0-10	Total (8-10): El acceso a las licencias es completo y sin restricciones. Todos los usuarios que necesitan acceder a la herramienta tienen una licencia disponible y pueden utilizarla plenamente. Parcial (4-7): Algunos usuarios pueden tener acceso completo, mientras que otros

			pueden tener acceso parcial o limitado a ciertas funciones o características. Nula (0-3): Ningún usuario tiene acceso a la herramienta debido a la falta de licencias disponibles o restricciones en su uso.
Costos	Bueno | Nulo	0-10	Bueno (6-10): Pago mensual menor o igual a 25 dólares. Nulo (0-5): Pago mensual mayor a 25 dólares.
Compatibilidad SO	Bueno | Regular | Nulo	0-10	Bueno (6-10): La herramienta es compatible con los sistemas operativos Windows, Linux, Mac OS. Nulo (0-5): No hay compatibilidad con el sistema operativo Windows, Linux, Mac OS.
Integraciones	Total | Parcial | Nula	0-10	Total (8-10): La herramienta permite integrar con más de 3 herramientas. Parcial (4-7): La herramienta permite integrar con 1 a 3 herramientas. Nula (0-3): No hay integraciones disponibles con otros sistemas, plataformas o herramientas.
Seguridad y protección de datos	Total | Parcial | Nula	0-10	Total (8-10): La herramienta proporciona un alto nivel de seguridad y protección de datos. Esto incluye medidas sólidas de cifrado, autenticación, control de acceso y políticas de privacidad bien definidas. Parcial (4-7): La herramienta ofrece cierto nivel de seguridad y protección de datos, pero puede haber algunas áreas de mejora o vulnerabilidades identificadas. Nula (0-3): Indica que no hay seguridad o protección de datos proporcionada por la herramienta.
Compatibilidad con lenguajes de programación	Total | Parcial | Nula	0-10	Total (8-10): Los lenguajes que se requieren sean compatibles incluyen, pero no se limitan a: Python, JavaScript, Java, C++, C#, Ruby, PHP, Swift, y Kotlin. Parcial (4-7): La herramienta tiene un nivel moderado de compatibilidad con algunos lenguajes de programación. Esto significa que soporta algunos, pero no todos los lenguajes principales. Los lenguajes que se requieren sean compatibles en este nivel incluyen al menos: Python, JavaScript, y Java. Nula (0-3): Indica que no hay compatibilidad con ningún lenguaje de programación. En este caso, la herramienta no soporta ninguno de

			los lenguajes de programación requeridos: Python, JavaScript, Java, C++, C#, Ruby, PHP, Swift, y Kotlin.
Curva de aprendizaje	Fácil | Moderada | Difícil	0-10	Fácil (8-10): La herramienta presenta una curva de aprendizaje fácil y rápida. La duración del aprendizaje inicial esta entre 0 y 3 créditos. Moderada (4-7): La herramienta tiene una curva de aprendizaje moderada. La duración del aprendizaje inicial esta entre 4 y 5 créditos. Difícil (0-3): La herramienta tiene una curva de aprendizaje muy empinada. La duración del aprendizaje inicial mayor a 5 créditos.
Interfaz de usuario	Bueno | Regular | Nulo	0-10	Bueno (8-10): La interfaz de usuario de la herramienta es altamente intuitiva, fácil de usar y estéticamente atractiva. Regular (4-7): La interfaz de usuario de la herramienta es funcional, carece de aspectos de usabilidad o diseño. Los usuarios pueden encontrar algunas áreas de la interfaz confusas o poco intuitivas. Nulo (0-3): Indica que la interfaz de usuario de la herramienta es deficiente o prácticamente inexistente.

Nota. Se relaciona las referencias que se aplican en la valoración a las herramientas de automatización de pruebas de software a evaluar. *Fuente*. Autoría Propia

Tabla 7

Evaluación de Herramientas.

Herramienta para Automatizar Pruebas de Software					
Aplicación	**Parámetro de Evaluación**	**Valor Cualitativo**	**Escala Cuantitativa**	**Total**	**URL**
Selenium Web Driver	Acceso a documentación técnica	Total	8	8	https://www.selenium.dev/documentation/
	Soporte de la herramienta	Regular	8	8	https://www.selenium.dev/support/

	(empresa-comunidad)				
	Licencias	Total	7	7	https://www.selenium.dev/
	Costos	Bueno	10	10	https://www.selenium.dev/
	Compatibilidad SO	Bueno	8	8	https://www.selenium.dev/downloads/
	Integraciones	Parcial	6	6	https://profile.es/blog/como-automatizar-pruebas-con-selenium/
	Seguridad y protección de datos	Nula	2	2	https://www.selenium.dev/
	Compatibilidad con lenguajes de programación	Total	9	9	https://www.selenium.dev/downloads/
	Curva de aprendizaje	Difícil	3	3	https://www.gartner.com/reviews/market/application-development-integration-and-management-others/vendor/selenium/product/selenium-webdriver https://www.capterra.com/p/234796/Selenium-IDE/reviews/
	Interfaz de usuario	Regular	6	6	https://www.gartner.com/reviews/market/application-development-integration-and-management-others/vendor/selenium/product/selenium-webdriver https://www.capterra.com/p/234796/Selenium-IDE/reviews/
Katalon	Acceso a documentación técnica	Parcial	7	7	https://docs.katalon.com/
	Soporte de la herramienta (empresa-comunidad)	Regular	5	5	https://docs.katalon.com/
	Licencias	Parcial	7	7	https://katalon.com/pricing
	Costos	Nula	3	3	https://katalon.com/pricing
	Compatibilidad SO	Bueno	9	9	https://katalon.com/download
	Integraciones	Bueno	8	8	https://katalon.com/integrations
	Seguridad y protección de datos		7	7	https://katalon.com/security
	Compatibilidad con lenguajes de programación	Parcial	7	7	https://jugnicaragua.org/katalon-para-iniciar-en-la-automatizacion-de-testcase/

	Curva de aprendizaje	Fácil	8	8	https://katalon.com/download
	Interfaz de usuario	Regular	7	7	https://www.gartner.com/reviews/market/ai-augmented-software-testing-tools/vendor/katalon/product/katalon
TestComplete	Acceso a documentación técnica	Regular	5	5	https://smartbear.com/product/testcomplete/
	Soporte de la herramienta (empresa-comunidad)	Regular	5	5	https://smartbear.com/product/testcomplete/
	Licencias	Parcial	7	7	https://smartbear.com/product/testcomplete/pricing/
	Costos	Nula	3	3	https://smartbear.com/product/testcomplete/pricing/
	Compatibilidad SO	Bueno	8	8	https://smartbear.com/product/testcomplete/features/
	Integraciones	Bueno	8	8	https://smartbear.com/product/testcomplete/integrations/
	Seguridad y protección de datos	Regular	7	7	https://support.smartbear.com/testcomplete/
	Compatibilidad con lenguajes de programación	Parcial	7	7	https://support.smartbear.com/testcomplete/docs/index.html
	Curva de aprendizaje	Fácil	8	8	https://support.smartbear.com/testcomplete/
	Interfaz de usuario	Regular	6	6	https://support.smartbear.com/testcomplete/
Cypress	Acceso a documentación técnica	Total	9	9	https://docs.cypress.io/
	Soporte de la herramienta (empresa-comunidad)	Bueno	9	9	https://www.cypress.io/support
	Licencias	Total	9	9	https://www.cypress.io/pricing
	Costos	Bueno	8	9	https://www.cypress.io/pricing
	Compatibilidad SO	Bueno	9	9	https://docs.cypress.io/guides/getting-started/installing-cypress(Consultada el 13/05/202024)
	Integraciones	Bueno	9	9	https://docs.cypress.io/plugins (Consultada el 12/05/202024)
	Seguridad y protección de datos	Total	8	8	https://www.cypress.io/security (Consultada el 12/05/202024)

Compatibilidad con lenguajes de programación	Total	9	9	https://docs.cypress.io/guides/overview/why-cypress (Consultada el 13/05/202024)
Curva de aprendizaje	Moderada	7	7	https://learn.cypress.io/ (Consultada el 13/05/202024)
Interfaz de usuario	Bueno	9	9	https://www.capterra.co/reviews/168276/cypress (Consultada el 14/05/202024)

Nota. Se detalla la evaluación aplicada a las herramientas para automatizar pruebas de software.

Fuente. Autoría Propia basada en la documentación de cada herramienta.

Tabla 8

Resumen de evaluación de herramientas para pruebas automatizadas de software

Herramienta	Acceso a documentación técnica	Soporte de la herramienta (empresa - comunidad)	Licencias	Costos	Compatibilidad SO	Integraciones	Seguridad y protección de datos	Compatibilidad con lenguajes de programación	Curva de aprendizaje	Interfaz de usuario	Promedio
Cypress	9	9	9	8	9	9	8	9	9	9	8,8
Selenium Web Driver	8	8	7	10	8	6	2	9	7	6	7,2
Katalon	7	5	7	3	9	8	7	7	7	7	6,7
TestComplete	5	5	7	3	8	8	7	7	6	6	5,7

Nota. Se realiza resumen de la evaluación realizada a las herramientas para automatizar pruebas de software. *Fuente*. Autoría Propia

Cypress:

Promedio: 8.8

Aspectos destacados: Excelente soporte de la herramienta y compatibilidad con múltiples lenguajes de programación.

Áreas de mejora: Costo y licencias podrían ser una limitación para algunos usuarios.

Selenium Web Driver:

Promedio: 7.2

Aspectos destacados: Buen acceso a la documentación técnica y compatibilidad con múltiples lenguajes de programación.

Áreas de mejora: Seguridad y protección de datos ni licencias propias.

Katalon:

Promedio: 6.7
Aspectos destacados: Buen soporte de la herramienta y compatibilidad con diferentes sistemas operativos.

Áreas de mejora: La curva de aprendizaje y la seguridad y protección de datos podrían mejorar.

TestComplete:

Promedio: 5.7

Aspectos destacados: Buena integración y seguridad, así como una curva de aprendizaje moderada.

Áreas de mejora: Acceso a la documentación técnica y licencias.

En general, Cypress destaca como la herramienta más completa y mejor valorada en la evaluación, seguida de cerca por Katalon.

Toma de Tiempos en Ejecución de Pruebas Automatizadas de Software

La tabla presentada a continuación constituye el resumen de la ejecución del plan de pruebas de una unidad de negocio, entendiendo por unidad de negocio a un módulo que hace parte de un proyecto de software más amplio. Esto módulos en algunos casos son trasversales, es decir que son usados por diferentes objetos de negocio, por ejemplo, la unidad de negocio de facturación.

El plan objeto de la muestra está compuesto por 15 pasos diseñados (Ver anexo 1), se llevó a cabo para evaluar con precisión la funcionalidad. Este módulo es fundamental para la realización de diversas acciones, dentro de la aplicación.

Tabla 9

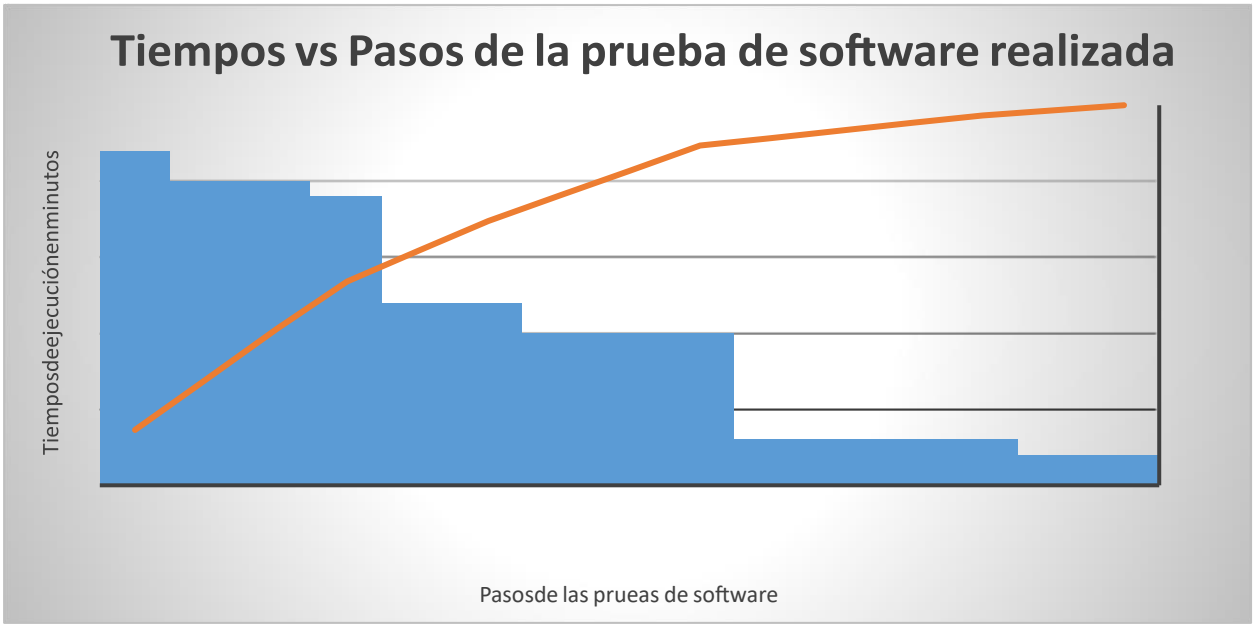
Resumen de Ejecución del Plan de Pruebas

Nombre del Test	Duración (min)	Observaciones
Paso 7	0,22	Retraso por ambiente de pruebas
Paso 9	0,2	Retraso en revisión de data y ambiente
Paso 11	0,2	Retraso en revisión de data y ambiente de pruebas
Paso 8	0,19	Retraso en revisión de data y ambiente de pruebas
Paso 1	0,12	Sin novedad
Paso 2	0,12	Sin novedad
Paso 13	0,1	Sin novedad
Paso 14	0,1	Sin novedad
Paso 15	0,1	Sin novedad
Paso 3	0,03	Sin novedad
Paso 4	0,03	Sin novedad
Paso 5	0,03	Sin novedad
Paso 10	0,03	Sin novedad
Paso 6	0,02	Sin novedad
Paso 12	0,02	Sin novedad
Total	1, 50	

Nota. Se realiza la relación de los tiempos tomados en cada paso realizado en la prueba ejecutada de forma automatizada. *Fuente*. Autoría Propia

Gráfica 2

Tiempos vs Pasos de la prueba de software realizada.



Nota. Este gráfico representa la relación de los tiempos empleados en cada paso de la prueba automatizada realizada. *Fuente*. Autoría Propia

La gráfica presenta una comparación detallada entre los tiempos de ejecución en minutos y los pasos de una prueba automatizada de software. En el eje horizontal se representan los diferentes pasos de la prueba, mientras que en el eje vertical izquierdo se indica el tiempo de ejecución en minutos para cada paso, y en el eje vertical derecho se muestra una escala de referencia adicional. Los pasos 7, 9, 11 y 8 tienen los tiempos de ejecución más altos, con 0,22, 0,20, 0,20 y 0,19 minutos respectivamente. En contraste, los pasos 4, 5, 10 y 12 tienen los tiempos más bajos, de 0,02 minutos, seguidos por los pasos 3 y 6 con 0,03 minutos. La línea naranja sugiere una acumulación del tiempo total o una tendencia promedio, indicando que los tiempos de ejecución disminuyen conforme avanza la prueba. Esta información es valiosa para identificar las áreas que consumen más tiempo y que podrían necesitar optimización, destacando

que los primeros pasos son los más críticos en términos de tiempo, mientras que los pasos finales son más eficientes.

Según la tabla número 8, de la toma de tiempos de los pasos realizados en el test de pruebas, se puede concluir lo siguiente:

El tiempo promedio de ejecución de una prueba automatizada es de aproximadamente 1,50 minutos por cada test. Siendo el máximo 0,22 minutos y el mínimo 0,02 minutos.

El paso siete tardo más debido a que varios campos tienen validación de data solicitando el doble ingreso de la información confirmando que la data coincida.

En condiciones normales un set de pruebas incluye en promedio de 20 test (un flujo completo) en un plan de pruebas para verificar el correcto funcionamiento de un componente. Este flujo puede ser el inicio de varios flujos de la aplicación. De acuerdo con esto, el tiempo total estimado es:

20 test × 1,50 minutos = 30 minutos

Dependiendo el módulo del software que se esté probando, la prueba completa puede contener uno o más flujos, entendiendo por flujo un set completo como se muestra en la tabla número 8. En promedio en el software desarrollado en la empresa objeto de este análisis está compuesto de 8 flujos. De tal manera que:

8 flujos x 30 minutos (set de pruebas) = 240 minutos equivalente a 4 horas En

horas ingeniero sería:

4 x $30.500* = $122.000. Una reducción de $170.800

Evaluación del Impacto de Pruebas Automatizadas en el Desarrollo de Software

La siguiente tabla presenta una evaluación del impacto de las pruebas automatizadas en el desarrollo de software, con los parámetros de velocidad de las pruebas, el costo por hora del ingeniero, la productividad del equipo y la satisfacción del equipo. Cada parámetro se clasifica cualitativamente y se asigna una escala cuantitativa para facilitar la interpretación de los resultados y comprender mejor los beneficios y desafíos asociados con la automatización de pruebas.

Tabla 10

Formato de evaluación del impacto de automatización

Impacto de las Pruebas Automatizadas en el Desarrollo de Software			
Parámetro de Evaluación	**Valor Cualitativo**	**Escala Cuantitativa**	**Interpretación**
Velocidad de las pruebas	Lenta | Moderada | Rápida	0-3	Lenta (1): Tiempo de ejecución mayor que las pruebas manuales. Moderada (2): Tiempo de ejecución similar al de las pruebas manuales. Rápida (3): Tiempo de ejecución menor que las pruebas manuales.
Costo hora ingeniero	Bajo | Moderado | Alto	0-3	Alto (1): Costo por hora del ingeniero es mayor. Moderado (2): Costo por hora del ingeniero es similar. Bajo (3): Costo por hora del ingeniero es menor.
Productividad del equipo	Baja | Moderada | Alta	0-3	Baja (1): Tiempo de entrega de pruebas finalizadas con retrasos. Moderada (2): Tiempo de entrega de pruebas finalizadas a tiempos límite de entrega. Alta (3): Tiempo de entrega de pruebas finalizadas antes del límite de entrega.
Satisfacción del equipo	Insatisfecho | Neutral Satisfecho	0-3	Insatisfecho (1): El equipo está descontento con las herramientas y procesos de automatización. Neutral (2): El equipo está ni satisfecho ni insatisfecho con las herramientas y procesos de automatización. Satisfecho (3): El equipo está contento con las herramientas y procesos de automatización.

Nota. Se relación valores de referencia para evaluación del impacto de automatización en el ciclo de desarrollo de software. *Fuente.* Autoría Propia

Gráfica 3

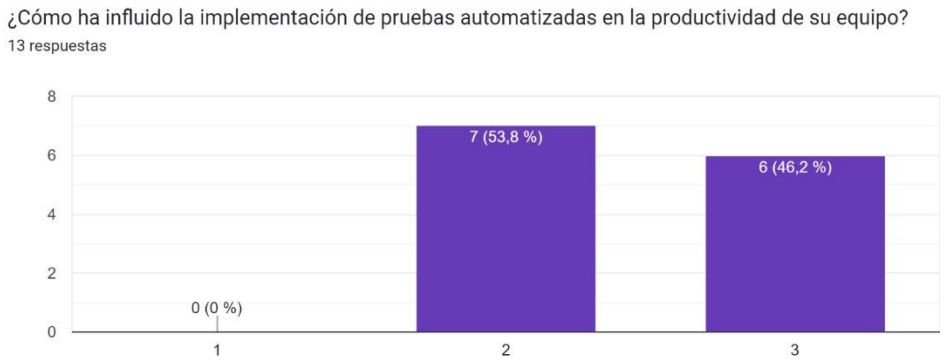
Calificación de la velocidad del proyecto

Nota. Gráfica que muestra la calificación de la velocidad de las pruebas automatizadas en el proyecto. *Fuente*. Autoría Propia

En los resultados de la encuesta la mayoría de los participantes (53.8%) tiene una percepción positiva respecto a las pruebas automatizadas impactando la velocidad, calificándolas con la puntuación más alta disponible. Esto indica que el proceso de pruebas automatizadas está funcionando bien para la mayoría.

Gráfica 4

Representación de la influencia de la automatización



Nota. Se gráfica las respuestas recibidas en la encuesta realizada sobre la influencia en la implementación de pruebas automatizadas en la productividad del equipo. *Fuente*. Autoría Propia.

En los resultados de la encuesta, la mayoría de los participantes (53,8%) indican que aún no se observa un impacto significativo de la automatización en la productividad del equipo. Sin embargo, un 46,2% considera que la automatización sí ha tenido un impacto positivo en la productividad.

Gráfica 5

Representación de la satisfacción en el equipo

Nota. Se gráfica las respuestas recibidas en la encuesta realizada. *Fuente.* Autoría Propia

La mayoría de los encuestados (53.8%) percibe un impacto muy positivo en su satisfacción gracias a las pruebas automatizadas. Esto sugiere que las pruebas automatizadas están ayudando al equipo a trabajar de manera más eficiente y efectiva, lo cual contribuye a una mayor satisfacción. Y el 46.2% de los encuestados que otorgaron una calificación de 2 representan un grupo que siente que, aunque hay mejoras en la satisfacción, existen áreas que podrían optimizarse para incrementar aún más la satisfacción del equipo.

A partir de los resultados de la encuesta, se puede concluir que las pruebas automatizadas de software tienen el potencial de impactar positivamente diversos aspectos del desarrollo de software. Entre estos aspectos destacan:

- Velocidad de las Pruebas: Las pruebas automatizadas permiten ejecutar casos de prueba de manera más rápida y eficiente, reduciendo el tiempo necesario para validar el código y detectar errores.

- Costo de Ingeniero: Al automatizar las pruebas repetitivas y manuales, los ingenieros pueden dedicar más tiempo a tareas de mayor valor, como el desarrollo de nuevas funcionalidades y la resolución de problemas complejos, optimizando así el costo y el uso del talento técnico.
- Productividad: La automatización de pruebas contribuye a aumentar la productividad del equipo al minimizar interrupciones y tiempos de espera, permitiendo un flujo de trabajo más continuo y ágil.
- Satisfacción del Equipo: La implementación de pruebas automatizadas puede mejorar la moral y satisfacción del equipo al reducir el estrés asociado con las pruebas manuales y proporcionar resultados más rápidos y confiables. Un equipo más satisfecho tiende a ser más motivado y eficiente.

Estos beneficios influyen directamente en el desarrollo de software de las siguientes maneras:

- Calidad del Software: Las pruebas automatizadas aseguran una mayor cobertura de pruebas y una detección temprana de errores, mejorando la calidad del producto final.
- Ciclos de Desarrollo más Cortos: Al reducir los tiempos de prueba, los ciclos de desarrollo pueden acortarse, permitiendo entregas más frecuentes y rápidas al mercado.
- Mejora Continua: La automatización facilita la integración y la entrega continuas (CI/CD), promoviendo una cultura de mejora continua en el equipo.

- Escalabilidad: A medida que los proyectos crecen, las pruebas automatizadas permiten escalar las actividades de prueba sin un aumento proporcional en el tiempo o los recursos necesarios.

Análisis de Costo-Beneficio Detallado usando Cypress

Costos Directos

- Horas Ingeniero:
  - Desarrollo de Pruebas:
    - Se estima 150 horas de un ingeniero para desarrollar los scripts de pruebas en Cypress.
    - Costo por hora de ingeniero: $30.500.
    - **Costo Total de Desarrollo:** 150 horas * $30.500/hora = **$4.575.000.** o Mantenimiento:
    - Se estima que el mantenimiento requerirá 15 horas mensuales.
    - Costo Anual de Mantenimiento: 15 horas/mes * 12 meses * $30.500/hora = $5.490.000.
- Licencias y Herramientas:
  - Cypress Dashboard (si se decide usar): $ 67 dólares /mes.
  - Costo Anual de Herramientas: $67 dólares/mes * 12 meses = $804 dólares año. $3.339.354 aprox. $3.340.000 puede variar el precio del dólar
- Infraestructura de CI/CD:
  - Se estima un costo adicional para ajustar la infraestructura de CI/CD: $1,500.000 (pago único).

Total, de Costos Directos Anuales: $4.575.000+ $5.490.000+$3.340.000 + $1.500.000= **$14.905.000**

Costos Indirectos

- Formación:
    - Se estima que la capacitación de 3 ingenieros requerirá 80 horas por persona para adquirir las habilidades necesarias en Cypress.
    - Costo Total de Capacitación: 3 ingenieros * 80 horas * $30.500/hora = $7.320.000.
- Integración con otras herramientas:
    - Se estima un costo de$ 1.000.000 para la integración de Cypress con el resto del entorno de pruebas.

Total, de Costos Indirectos: 7.320.000 + $1.000.000 = **$8.320.000 Beneficios**

- Ahorro de Tiempo:
    - Se estima que la automatización reducirá el tiempo de ejecución de pruebas en un 80%, ahorrando aproximadamente 1005 horas al año. ○ Valor del

    Ahorro en Tiempo: 1005 horas * $30.500/hora = $30.652.500.

- Reducción de Errores:
    - La reducción de errores en producción podría ahorrar a la empresa $1.500.000 anuales en costos de corrección y soporte.

Total, de Beneficios Anuales: $30.652.500 + $1.500.000 = **$32.152.500**

Retorno de Inversión (ROI)

El ROI se calcula de la siguiente manera:

$$ROI = \frac{Beneficios - Costos\ Directos\ e\ Indirectos}{Costos\ Directos\ e\ Indirectos}$$

- Costos Totales: $14.905.000 (Directos) + $8.320.000 (Indirectos) = $23.225.000
- Beneficios Totales: $32.152.500

$$ROI = \frac{32.152.500 - 23.225.000}{23.225.000}$$

$$ROI = \frac{8.927.500}{23.225.000}$$

$$ROI = \frac{8.927.500}{23.225.000} = 0{,}38 = 38\%$$

El ROI es positivo (38%), lo que indica que, según estos valores, la implementación sería económicamente viable.

Aclaración: Los valores presentados son estimaciones aproximadas y se utilizan únicamente como referencia. Los costos y beneficios reales pueden variar según las circunstancias específicas del proyecto, la configuración de herramientas, y otros factores contextuales. Es importante ajustar estos números con base en datos concretos y actualizados a medida que se avance en el análisis.

Evaluación del Proceso de Automatización de Pruebas de Software en la Organización: Aplicación de Estándares de Calidad Definidos

En el apartado anterior, se realizó un diagnóstico de la empresa en el proceso de pruebas de software. Se llevó a cabo una toma de tiempos en pruebas manuales, lo que permitió comparar estas con las pruebas automatizadas utilizando la herramienta seleccionada, basada en los

resultados de la evaluación realizada. Ahora, es momento de revisar y resumir brevemente las actividades realizadas hasta la fecha.

Evaluación de Herramientas para Automatizar Pruebas de Software

La evaluación de herramientas para automatizar pruebas de software requiere considerar múltiples variables que afectan su usabilidad, efectividad y costo. A continuación, se justifica la selección de cada variable incluida en la evaluación:

Tabla 11

Representación de cada variable seleccionada

Parámetro	Importancia	Evaluación
Acceso a Documentación Técnica	Crucial para la comprensión y uso eficiente de la herramienta.	Escala cualitativa y cuantitativa que mide la accesibilidad y exhaustividad de la documentación.
Soporte de la Herramienta (EmpresaComunidad)	Asegura ayuda cuando se enfrentan problemas. Incluye soporte de la empresa y la comunidad de usuarios.	Escala que refleja la velocidad y utilidad de las respuestas y la colaboración entre la empresa y la comunidad.
Licencias	Determina cuántos usuarios pueden utilizar la herramienta sin restricciones.	Evaluación de la disponibilidad de licencias (acceso completo, parcial o nulo).
Costos	Crítico, especialmente para empresas con presupuestos limitados.	Escala que compara el costo con otras herramientas.
Compatibilidad con Sistemas Operativos (SO)	Asegura que la herramienta pueda ser utilizada en diferentes entornos de desarrollo y pruebas.	Medida de compatibilidad con Windows, Linux y Mac OS.
Integraciones	Permite automatizar más procesos y reducir la intervención manual.	Número de integraciones disponibles y destacando aquellas con múltiples sistemas.

Seguridad y Protección de Datos	Ofrece medidas robustas para proteger la información sensible.	Medida que protege la información sensible en un entorno donde la seguridad de los datos es primordial.
Compatibilidad con Lenguajes de Programación	Permite utilizar la herramienta en diferentes proyectos y entornos.	Medida que determina la versatilidad y adaptabilidad de la herramienta.
Curva de Aprendizaje	Permite a los usuarios comenzar a usar la herramienta rápidamente.	Medida que impacta la adopción efectiva de la herramienta por parte de los usuarios.
Interfaz de Usuario	Mejora la experiencia del usuario y facilita la utilización de la herramienta.	Evaluación de la calidad de la interfaz en términos de usabilidad y diseño estético.

Nota. Se representa los parámetros usados para evaluar las herramientas de automatización.

Fuente. Autoría Propia

Al evaluar las herramientas de pruebas automatizadas, Cypress se destaca con un promedio de 8.8 debido a su excelente soporte y compatibilidad con múltiples lenguajes de programación. No obstante, el costo y las licencias pueden ser limitantes para algunos usuarios. Selenium Web Driver obtiene un promedio de 7.2, sobresaliendo por su buen acceso a la documentación técnica y compatibilidad con varios lenguajes de programación, aunque presenta áreas de mejora en seguridad y protección de datos, y carece de licencias propias. Katalon, con un promedio de 6.7, es apreciado por su buen soporte y compatibilidad con diferentes sistemas operativos, pero tiene una curva de aprendizaje pronunciada y aspectos de seguridad que necesitan atención. Finalmente, Test Complete, con un promedio de 5.7, ofrece buena integración

y seguridad junto con una curva de aprendizaje moderada, pero requiere mejorar el acceso a la documentación técnica y la gestión de licencias.

En general, Cypress destaca como la herramienta más completa y mejor valorada en la evaluación, seguida de cerca por Katalon.

Comparación de Tiempos y Costos: Pruebas Manuales vs. Pruebas Automatizadas

Pruebas Manuales

El análisis del gráfico N°1."Tiempos vs Pasos de la prueba de software" revela que los pasos 1, 8, 9 y 11 tienen tiempos de ejecución más altos, lo que sugiere áreas críticas para optimizar la eficiencia. Con un tiempo promedio de 3,6 minutos por prueba manual y un máximo de 0,6 minutos, el paso 1 es el más demorado debido a la latencia del ambiente y las demoras en la consulta de datos de factura. Un set de pruebas típico incluye 20 tests, totalizando 72 minutos por flujo y 576 minutos (9,6 horas) para un módulo con 8 flujos, lo que se traduce en un costo de $292.800 en horas ingeniero. Para mejorar, se recomienda enfocarse en reducir los tiempos de los pasos más largos.

Pruebas Automatizadas

El análisis del gráfico N° 2."Tiempos vs Pasos de la prueba de software" compara los tiempos de ejecución en minutos de los diferentes pasos de una prueba automatizada de software, mostrando que los pasos 7, 9, 11 y 8 tienen los tiempos más altos, con 0,22, 0,20, 0,20 y 0,19 minutos respectivamente, mientras que los pasos 4, 5, 10 y 12 son los más rápidos con 0,02 minutos. La línea naranja sugiere que los tiempos de ejecución disminuyen conforme avanza la prueba. El tiempo promedio de ejecución de una prueba automatizada es de 1,50 minutos, con un máximo de 0,22 minutos. El paso 7 tarda más debido a la validación doble de datos. Un set de 20

tests requiere 30 minutos, y para un módulo con 8 flujos, el total es de 240 minutos (4 horas), resultando en un costo de $122.000, una reducción de $170.800 comparado con el costo anterior.

Evaluación del Proceso de Automatización de Pruebas de Software Aplicando Estándares de ***Calidad***

El cuadro de evaluación que se presenta a continuación está basado en el estándar ISO 25010, que se enfoca en la calidad del software. Este estándar abarca diversas categorías y subcategorías que permiten una evaluación cualitativa y cuantitativa de las características del software. A continuación, se describen las categorías evaluadas junto con sus respectivas subcategorías, los valores cualitativos asignados, la escala cuantitativa utilizada y la interpretación correspondiente.

Tabla 12

Formato de categorías según ISO 25010 para evaluar el proceso de automatización de pruebas de software

Categoría ISO 25010	**Subcategorías**	**Valor Cualitativo**			**Escala Cuantitativa**	**Interpretación**
Mantenibilidad	Modularidad, Reusabilidad	Bajo	Moderado	Alto	1-3	1 (Bajo): El software tiene baja modularidad y reusabilidad, lo que dificulta las tareas de mantenimiento y actualización. 2 (Moderado): El software tiene una estructura moderadamente modular y reutilizable, permitiendo algunas tareas de

						mantenimiento con relativa facilidad. 3 (Alto): El software es altamente modular y reutilizable, facilitando significativamente el mantenimiento y la actualización.
Compatibilidad	Coexistencia, Portabilidad	Limitada	Compatible	Totalmente Compatible	1-3	1 (Limitada/Baja): El software tiene limitadas capacidades de coexistencia y portabilidad, restringiendo su uso en diferentes entornos y plataformas. 2 (Compatible/Moderada): El software es moderadamente compatible y portable, funcionando bien en varios entornos y plataformas, pero con algunas restricciones. 3 (Totalmente Compatible/Alta): El software es altamente compatible y portable, operando sin problemas en una amplia variedad de entornos y plataformas.
No directamente evaluado	Relacionado con Eficiencia de Recursos y Mantenibilidad	Alto	Moderado	Bajo	1-3	1 (Alto): Alta eficiencia de recursos y mantenibilidad, sugiriendo un rendimiento óptimo y fácil mantenimiento. 2 (Moderado): Eficiencia de recursos y mantenibilidad moderada, con un rendimiento aceptable y mantenimiento

						posible pero no óptimo. 3 (Bajo): Baja eficiencia de recursos y mantenibilidad, indicando un rendimiento deficiente y dificultades en el mantenimiento.
Compatibilidad	Coexistencia, Portabilidad	Baja	Moderada	Alta	1-3	1 (Baja): El software tiene baja interoperabilidad, limitando su capacidad para interactuar con otros sistemas o componentes. 2 (Moderada): El software tiene interoperabilidad moderada, pudiendo interactuar con algunos sistemas o componentes, pero con ciertas limitaciones. 3 (Alta): El software es altamente interoperable, permitiendo una interacción fluida con una amplia variedad de sistemas y componentes.
Compatibilidad	Interoperabilidad	Baja	Moderada	Alta	1-3	1 (Baja): El software tiene bajos niveles de seguridad, con riesgos significativos para la confidencialidad, integridad y autenticidad de los datos. 2 (Moderada): El software ofrece seguridad moderada, con algunas medidas de protección en su lugar, pero con áreas de vulnerabilidad.

						3 (Alta): El software proporciona altos niveles de seguridad, con medidas robustas que protegen la confidencialidad, integridad y autenticidad de los datos.
Seguridad	Confidencialidad, Integridad, No repudio, Responsabilidad, Autenticidad	Baja	Moderada	Alta	1-3	1 (Baja): El software tiene bajos niveles de seguridad, con riesgos significativos para la confidencialidad, integridad y autenticidad de los datos. 2 (Moderada): El software ofrece seguridad moderada, con algunas medidas de protección en su lugar, pero con áreas de vulnerabilidad. 3 (Alta): El software proporciona altos niveles de seguridad, con medidas robustas que protegen la confidencialidad, integridad y autenticidad de los datos.
Compatibilidad	Coexistencia, Portabilidad	Baja	Moderada	Alta	1-3	1 (Baja): El software tiene baja interoperabilidad, limitando su capacidad para interactuar con otros sistemas o componentes. 2 (Moderada): El software tiene interoperabilidad moderada, pudiendo interactuar con algunos sistemas o

						componentes, pero con ciertas limitaciones. 3 (Alta): El software es altamente interoperable, permitiendo una interacción fluida con una amplia variedad de sistemas y componentes.
Usabilidad	Aprendizaje, Operabilidad, Comprensibilidad, Atractividad	Poco Intuitiva	Moderadamente Intuitiva	Muy Intuitiva	1-3	1 (Alta/Poco Intuitiva): La usabilidad es alta, indicando que el aprendizaje es rápido y fácil; sin embargo, si se evalúa como poco intuitiva, su interfaz puede ser difícil de usar. 2 (Moderada/Moderadamente Intuitiva): El aprendizaje y la operabilidad son moderados, requiriendo algún esfuerzo para dominar, pero generalmente comprensible. 3 (Baja/Muy Intuitiva): La usabilidad es baja en términos de dificultad de aprendizaje, con una interfaz de usuario muy intuitiva y fácil de usar.

Nota. Se realiza cuadro de referencia para aplicar evaluación según norma ISO 25010 al proceso de automatización de pruebas de software. *Fuente*. Norma ISO 251010

Tabla 13

Evaluación de la calidad del software utilizado para la automatización de pruebas de software según estándares de calidad

Categoría ISO 25010	Subcategorías	Valor Cualitativo	Escala Cuantitativa	TOTAL
Mantenibilidad	Modularidad, Reusabilidad	Alto	2	2
Compatibilidad	Coexistencia, Portabilidad	Totalmente Compatible	3	3
No directamente evaluado	Relacionado con Eficiencia de Recursos y Mantenibilidad	Alto	3	3
Seguridad	Confidencialidad, Integridad, No repudio, Responsabilidad, Autenticidad	Moderada	2	2
Compatibilidad	Coexistencia, Portabilidad	Alta	3	3
Usabilidad	Aprendizaje, Operabilidad, Comprensibilidad, Atractividad	Muy Intuitiva	3	3

Nota. Evaluación del proceso de automatización de pruebas de software según estándares de calidad ISO 25010. *Fuente*. Norma ISO 251010

Nota: La evaluación cuantitativa de las diferentes características del software se realiza utilizando una escala de 1 a 3, donde 1 representa el nivel más bajo o menos favorable, 2 indica un nivel moderado, y 3 corresponde al nivel más alto o favorable. Esta escala permite una valoración objetiva de cada subcategoría, facilitando la comparación y análisis de las distintas dimensiones de calidad del software.

El promedio obtenido es 2,6 en una escala de 3, lo que indica un desempeño cercano al nivel máximo posible.

Análisis de la Tabla

Mantenibilidad

Subcategorías: Modularidad, Reusabilidad

Valor Cualitativo: Alto

Escala Cuantitativa: 2

Interpretación: La mantenibilidad del sistema es alta, lo cual sugiere que el sistema está diseñado para ser fácilmente modificable y reutilizable.

Compatibilidad (Primera aparición)

Subcategorías: Coexistencia, Portabilidad

Valor Cualitativo: Totalmente Compatible

Escala Cuantitativa: 3

Interpretación: El sistema es completamente compatible con otros sistemas, lo que facilita la coexistencia y portabilidad sin problemas.

No directamente evaluado

Subcategorías: Relacionado con Eficiencia de Recursos y Mantenibilidad

Valor Cualitativo: Alto

Escala Cuantitativa: 3

Interpretación: Aunque no está directamente evaluado, el sistema muestra una alta eficiencia en términos de recursos y mantenibilidad, lo que sugiere un buen diseño en estas áreas.

Seguridad

Subcategorías: Confidencialidad, Integridad, No repudio, Responsabilidad, Autenticidad

Valor Cualitativo: Moderada

Escala Cuantitativa: 2

Interpretación: La seguridad del sistema es moderada, lo que indica que hay áreas de mejora en términos de confidencialidad, integridad, autenticidad y otros aspectos de seguridad.

Compatibilidad (Segunda aparición)

Subcategorías: Coexistencia, Portabilidad

Valor Cualitativo: Alta

Escala Cuantitativa: 3

Interpretación: Similar a la primera evaluación de compatibilidad, el sistema sigue demostrando una alta capacidad de coexistencia y portabilidad.

Usabilidad

Subcategorías: Aprendizaje, Operabilidad, Comprensibilidad, Atractividad

Valor Cualitativo: Muy Intuitiva

Escala Cuantitativa: 3

Interpretación: La usabilidad del sistema es muy intuitiva, lo que facilita el aprendizaje y la operabilidad para los usuarios.

La tabla presenta un análisis cualitativo y cuantitativo de diversas categorías basadas en el estándar ISO 25010, lo que proporciona una visión completa de las características y calidad del sistema evaluado.

Puntos Fuertes:

Compatibilidad: El sistema es totalmente compatible y altamente portable, lo que facilita su integración con otros sistemas y su uso en diferentes plataformas.

Usabilidad: La alta usabilidad asegura que los usuarios puedan aprender y operar el sistema de manera intuitiva.

Mantenibilidad y Eficiencia de Recursos: Ambas categorías muestran altos valores cualitativos y cuantitativos, lo que indica que el sistema es fácil de mantener y eficiente en el uso de recursos.

Áreas de Mejora:

Seguridad: La seguridad es moderada, sugiriendo que hay espacio para mejorar en aspectos críticos como la confidencialidad, integridad y autenticidad.

En general, el sistema muestra un buen rendimiento en la mayoría de las categorías, especialmente en compatibilidad y usabilidad, lo cual son cruciales para la aceptación y operabilidad del sistema. Sin embargo, se recomienda un enfoque particular

en mejorar la seguridad para garantizar la protección de datos y la confianza de los usuarios.

Conclusiones

El diagnóstico identificó la ausencia de diagramas de flujo y mapas de flujo de valor, elementos clave para comprender y optimizar el estado actual de las pruebas de software durante el ciclo de desarrollo. Además, la falta de pruebas automatizadas y la falta de una herramienta seleccionada para este fin resalta la necesidad de avanzar en la implementación de procesos más eficientes y estandarizados.

La aceptación por parte del equipo de la herramienta seleccionada para la implementación de pruebas de software automatizadas facilitó su integración en las tareas diarias. Sin embargo, fue necesario un ajuste inicial en la disponibilidad de tiempo, ya que la configuración y aprendizaje de la nueva herramienta requieren dedicación adicional en las etapas iniciales. El equipo mostró un progreso gradual en su adaptación a la herramienta, evidenciado por una mejora continua en la curva de aprendizaje.

La experiencia general del equipo con la herramienta fue positiva. Los beneficios de la automatización de pruebas se hicieron evidentes rápidamente, ya que permitió una reducción significativa en el tiempo necesario para ejecutar pruebas repetitivas.

La evaluación mostró que la implementación de pruebas automatizadas no solo reduce drásticamente el tiempo necesario para completar las pruebas, sino que también disminuye los costos asociados, logrando una reducción de casi un 58% en comparación con las pruebas manuales. Mientras que las pruebas manuales de un módulo con 8 flujos pueden tardar 9,6 horas, las pruebas automatizadas pueden completarse en solo 4 horas. Esta mejora en la eficiencia permite a los equipos de desarrollo concentrarse en tareas más estratégicas y de mayor valor, dejando las pruebas repetitivas y propensas a errores en manos de la automatización, que las maneja de manera más efectiva.

Además, al evaluar la herramienta utilizada y su impacto en el equipo, se encontró que, según los criterios establecidos por la ISO 25010, la herramienta seleccionada cumple con los requisitos apropiados, siendo adaptable al equipo y amigable en su uso por parte del personal.

Recomendaciones

Capacitación y Adopción

Capacitación Continua del Personal: Implementar programas de capacitación regulares para el equipo de desarrollo y pruebas sobre nuevas tecnologías y metodologías en automatización de pruebas.

Adopción de Nuevas Tecnologías: Establecer un proceso de evaluación y adopción de buenas prácticas y nuevas tecnologías en el proceso de automatización que puedan mejorar la eficiencia y la calidad del proceso de pruebas.

Definición de Métricas de Calidad

Definir Métricas de Calidad: Crear un conjunto de métricas claras y específicas para evaluar el impacto de la automatización de pruebas.

Impacto en el Ciclo de Vida del Desarrollo de Software

Mayor Eficiencia: Automatizar pruebas repetitivas y de regresión para reducir el esfuerzo manual y liberar recursos para otras tareas críticas del desarrollo, permitirá aumentar la eficiencia del equipo de desarrollo y permite una mejor utilización de los recursos disponibles.

Mejor Calidad: Implementar pruebas automatizadas en todas las etapas del ciclo de desarrollo, desde las pruebas unitarias hasta las pruebas de integración y de sistema.

Lanzamientos Más Rápidos: Integrar la automatización de pruebas en el proceso de integración y entrega continuas (CI/CD) para agilizar los lanzamientos de software.

Mayor Satisfacción del Cliente: Enfocar los esfuerzos de automatización en áreas críticas que impacten directamente en la experiencia del usuario final.

Estrategia General

Estrategia de Implementación: Desarrollar una estrategia de implementación escalonada para la automatización de pruebas, comenzando con proyectos piloto y expandiendo gradualmente la automatización a otros proyectos.

Referencias Bibliográficas

Al-Kilidar, H., Cox, K. G., & Kitchenham, B. (2005). The use and usefulness of the ISO/IEC 9126 quality standard. Proceedings of the 2005 International Symposium on Empirical Software Engineering. https://doi.org/10.1109/isese.2005.1541821.

Andreu, I. (2023, febrero 22). Lean manufacturing: ¿Qué es y cuáles son sus principios? APD España. https://www.apd.es/lean-manufacturing-que-es/

Asociación Colombiana de Ingenieros de Sistemas. (2022). El próximo salto de la industria TI en Colombia estará jalonado por retail y banca. ACIS. https://acis.org.co/portal/content/el-pr%C3%B3ximo-salto-de-la-industria-ti-encolombia-estar%C3%A1-jalonado-por-retail-y-banca-0#:~:text=De%20acuerdo%20con%20cifras%20del,a%20%24641%20mil%20en%202021.

Atlassian. (2023). Pruebas de software automatizadas para la entrega continua. Atlassian. https://www.atlassian.com/es/continuous-delivery/software-testing/automated-testing.

Centro de investigación especializado en el sector de software y sus relacionados. (2021). Gira regional 2021. Fedesoft. https://cenisoft.org/infografia-gira-regional-2021/

Colorado, L. P. (2020). Automatización de pruebas funcionales, un complemento para la calidad del software. Repositorio Institucional. http://hdl.handle.net/10654/37769.

David, R., Profesor, R., Marulanda López, J., & De Ingeniera, E. (2014). [Título del documento]. Recuperado de https://repository.eafit.edu.co/bitstream/handle/10784/5270/Jaime_MarulandaLopez_2014.pdf?sequence=2&isAllowed=y.

Diéguez, M., & Cares, C. (2011). De la gestión de seguridad en el ciclo de vida del software. En International Workshop on Software Engineering, Jornadas Chilenas de Computación.

Fernández, A. (2020). Ingresos del mercado de software para empresas en el mundo desde 2016 hasta 2021. Statista. https://es.statista.com/estadisticas/966415/ingresos-delmercado-de-software-para-empresas-en-el-mundo/

Fresno Chaves, C. (2019). Metodología de la investigación: Así de fácil. El Cid Editor.

Hernández Sampieri, R. H., Fernández Collado, C., & Baptista, M. P. (2014). Metodología de la investigación (6ª ed.). McGraw-Hill Education. https://www.uca.ac.cr/wpcontent/uploads/2017/10/Investigacion.pdf.

Ilimit. (2021, 19 de febrero). Beneficios de la automatización de pruebas. Ilimit. https://www.ilimit.com/blog/beneficios-automatizacion-pruebas/

Katalon. (s. f.). Business outcomes | Software testing & quality management. Katalon. https://katalon.com/business-outcomes.

Katalon. (s. f.-b). Quick guide for billing managers | Katalon docs. Katalon Docs. https://docs.katalon.com/docs/katalon-platform/get-started/onboarding-katalonplatform/quick-guide-for-billing-managers

Katalon. (s. f.). Quick guide for billing managers | Katalon docs. Katalon Docs. https://docs.katalon.com/docs/katalon-platform/get-started/onboarding-katalonplatform/quick-guide-for-billing-managers

Laoyan, S. (2024, febrero 10). Todo lo que necesitas saber sobre Six Sigma [2024]. Asana. https://asana.com/es/resources/six-sigma

Lerma González, H. (2009). Metodología de la investigación: Propuesta, anteproyecto y proyecto (4ª ed.). Ecoe Ediciones. https://elibronet.bibliotecavirtual.unad.edu.co/es/ereader/unad/69092?page=51

Morales Vindas, D., Chaves Rodríguez, J., Brizuela Hernández, R., & Retana, J. (n.d.). Los beneficios de la automatización para aumentar la calidad de pruebas de software. Recuperado el 7 de mayo de 2023, de http://44.209.83.190/bitstream/handle/123456789/10583/REF-1640202443-2.pdf?sequence=2&isAllowed=y

Pérez, D. A. (2015). Estudio comparativo de herramientas para la automatización de pruebas software [Tesis de grado, Universidad Carlos III de Madrid]. Repositorio UC3. https://earchivo.uc3m.es/bitstream/handle/10016/26573/PFC_Daniel_Alvaro_Perez.pdf

Saravanan, K., & Balakrishnan, S. K. (2019). Key factors & features influencing selection of open source functional test automation tools. International Journal of Recent Technology and Engineering, 8(2 Special Issue 10), 840-843. https://www.scopus.com.bibliotecavirtual.unad.edu.co/record/display.uri?eid=2-s2.0-85073575297&origin=resultslist&sort=plff&src=s&st1=Key+factors+%26+features+influencing+selection+of+open+source+functional+test+automation+tools&sid=13a420b891954810b19bdfbba724e07d&sot=b&sdt=b&sl=107&s=TITLE-ABS-KEY%28Key+factors+%26+features+influencing+selection+of+open+source+functional+test+automation+tools%29&relpos=0&citeCnt=1&searchTerm=

Serna López, G. A., Jaramillo, L. M., Upegui Giraldo, G., Gómez, R., & Bueno, Y. (2020). Piloto de automatización de pruebas de software en la Unidad de Servicios Tecnológicos del Centro de Servicios y Gestión Empresarial (CESGE) del SENA. Revista CINTEX, 25(2), 45-50. https://bibliotecavirtual.unad.edu.co/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edsair&AN=edsair.doi.....9ec02165439ca3b08728c335824d2ec4&lang=es&site=eds-live&scope=site

Serna, M., Martínez, R., & Tamayo, P. (2019). Una revisión a la realidad de la automatización de las pruebas del software. Computación y Sistemas, 23(1), 169-183. https://doi.org/10.13053/cys-23-1-2782

Tijerina Acosta, J. I. (1999). Benchmarking: Metodología de desarrollo y aplicación (Tesis doctoral, Universidad Autónoma de Nuevo León).

Trentia. (2022, 11 enero). Selenium 3 y 4: Ventajas, novedades y usos. Trentia. https://www.trentia.net/selenium-3-y-4-ventajas-novedades-y-usos/

Universidad Nacional Abierta y a Distancia UNAD. (2014). Acuerdo 006 del 28 de mayo de 2014. https://sgeneral.unad.edu.co/images/documentos/consejoAcademico/acuerdos/2014/COAC_ACUE_20140528_006.pdf

Velasco Tanguila, G. E. (2019). Análisis comparativo de herramientas para la automatización de pruebas de software [Tesis de grado, Universidad Central del Ecuador]. Repositorio UCE. http://www.dspace.uce.edu.ec/handle/25000/17868

Apéndices

Apéndice A

Plan de pruebas de software para funcionalidad.

Plan de pruebas de software para funcionalidad.

Nombre del módulo	
Fecha:	DD/MM/AAAA
Autor:	

Descripción del módulo: Espacio para ingresar los datos del usuario y cargos aplicados a la atención realizada.

Alcance de las pruebas: Se debe probar el detalle de la factura sea exitoso.

Casos de prueba:

Paso	Descripc ión	Pasos	Datos de entrada	Resultado esperado	Tiempo en minutos	Estado	Observacio nes
1	Ingresar al modulo	Iniciar sesión Selecciona r el rol Dar clic en Tareas de auditoría y luego el módulo facturas mirror.	User – password	Observar el módulo sin novedad			
2	Buscar una Prestador	Selecciona r una empresa Digitar el numero Dar clic en buscar	NIT	Observar el prestador consultado			

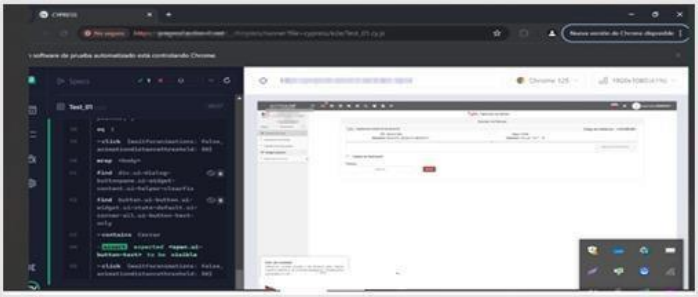
3	Clic en el prestador	Dar clic en el prestador deseado		Selección sin novedad			
4	Ingresar factura	Digitar el número de factura Dar clic en buscar	# factura	Debe aparecer información de la factura como (Factura, IPS, valor,fecha de radicación, tipo, botón editar, botón adjuntos y comentarios)			
5	Editar factura	Dar clic en el lápiz		Observar la data de la factura para editar y agregar información			
6	Informac ión de factura	Dar clic en editar		Observar data de radicación (número de factura, código de barras, fecha de radicación y fecha de factura) y opción de ingresar valores			
7	Ingresar valores	Ingrese valores (bruto, copago)	Valor	Verifica la data y el total			
8	Guardar valores	Dar clic en guardar		La data debe quedar guardada y enviar a la siguiente sección			

9	Datos del paciente	Seleccionar el tipo de documento Digitar el número de documento Tabular	Cc -# documento	Observar resultados si están y el botón datos no coinciden			
10	Datos del paciente	Dar clic en datos no coinciden		Observar formulario			
		Diligenciar datos					
11	Datos del paciente	Dar clic en agregar paciente Diligenciar de nuevo fechas Dar clic en agregar paciente	Nombres y apellidos	Observar siguiente sección			
12	Detalles	Ingresar el código del detalle Tabular Ingresar cantidad y costo	Código servicio	Ver nombre y totales			
13	Detalles	Dar clic en agregar detalle		Observar el detalle ingresado			
14	Paciente	Dar clic en guardar paciente		Observar datos agregados			
15	Finalizar	Dar clic en Terminar proceso Dar clic en mensaje de confirmaci ón		Observar mensaje de confirmación. Observar mensaje de factura se cargó con éxito			

Nota. Descripción de los pasos a realizar en el plan de pruebas. *Fuente*. Autoría Propia

Apéndice B

Ejecución de pruebas automatizadas.



Nota. Evidencia de ejecución de pruebas automatizadas. *Fuente*. Autoría Propia