DE LAS PRUEBAS MANUALES A LAS PRUEBAS AUTOMÁTICAS EN EL DESARROLLO ÁGIL DE SOFTWARE: CASO DE ESTUDIO UNIVERSIDAD DE NARIÑO

CALPA BRAVO ADRIANA CAROLINA

GARZÓN MORA JEFERSON STEVEN

PROGRAMA DE INGENIERÍA DE SISTEMAS
DEPARTAMENTO DE SISTEMAS
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE NARIÑO
AGOSTO, 2022

DE LAS PRUEBAS MANUALES A LAS PRUEBAS AUTOMÁTICAS EN EL DESARROLLO ÁGIL DE SOFTWARE: CASO DE ESTUDIO UNIVERSIDAD DE NARIÑO

Autores:
CALPA BRAVO ADRIANA CAROLINA, adrianacalpab@gmail.com
GARZÓN MORA JEFERSON STEVEN, jefersong5854@gmail.com

Informe final de trabajo de grado presentado como requisito para optar al título de Ingeniero de Sistemas, en modalidad investigación.

Director
HERNÁNDEZ GIOVANNI ALBERIO

PROGRAMA DE INGENIERÍA DE SISTEMAS
DEPARTAMENTO DE SISTEMAS
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE NARIÑO
AGOSTO, 2022

Nota exclusión de responsabilidad intelectual

“Las ideas y conclusiones aportadas en este Trabajo de Grado son responsabilidad de los autores”.

Artículo 1° del Acuerdo No. 324 de octubre 11 de 1966, emanado del Honorable Consejo Directivo de la Universidad de Nariño.

Nota de Aceptación

_____
_____
_____
_____
_____
_____

_____
Firma del director

_____
Firma del jurado evaluador

_____
Firma del jurado evaluador

_____
San Juan de Pasto, agosto de 2022

RESUMEN

El principal objetivo de esta investigación es aportar un proceso de desarrollo de pruebas automáticas de unidad y componentes, para los trabajos de grado del Programa de Ingeniería de Sistemas de la Universidad de Nariño que involucran la construcción de software, con el fin de garantizar la calidad de los mismos; además, de facilitar la ejecución de la fase de pruebas.

El enfoque para esta investigación es empírico-analítico, de tipo descriptivo. Como primer paso, se caracteriza la forma de aplicación de pruebas de los estudiantes en sus proyectos de grado entre los años 2010 a 2020, encontrando un bajo porcentaje de pruebas unitarias y sin evidencias del uso de pruebas de componentes.

Como resultado de esta investigación, se elabora un proceso de apoyo a la fase de pruebas de forma automática en Scrum, enfocado en pruebas de unidad y componentes, el cual puede ser aplicado en trabajos de grado. Este proceso hace uso de herramientas de automatización, gestión de no conformidades, artefactos para la especificación de historias de usuario, escenarios y casos de prueba. Se realiza una validación con 30 estudiantes, de quienes se recibe alto nivel de aceptación y una retroalimentación que permite identificar la necesidad de crear un proceso complementario para la gestión de actividades de Scrum.

Palabras Clave: *Automatic Test, Manual Test, Unit And Component Test, Agile Software Development*.

ABSTRACT

The main object of this investigation is contribute with a process of develop of unit and component automatic tests, of degree works of Systems Engineer Program of University of Nariño, that involve the software building, with the objetive to garantice quality and facilitate the execution of fase test.

The focus for this investigation is empiric-analitic, of descriptive kind. The fist step, is characterize the way of aplicate the test in the students degree proyects, that were develpment from 2010 to 2020, finding a low porcentaje of unit tests and without evidences of component tests.

The result of this investigation, was building a support process for the automatic test phase using Scrum, focused in unity and component tests, it can aplicate in degree works. This process uses automatic tests tools, management of failures tools, artifacts for user story specifications, scenarios and test case. It is validate with 30 students, who give a high leve lof acceptance, and a feedback with wich be identificate the need of create a complementary process, for the activities management of Scrum.

Keywords: *Automatic Test, Manual Test, Unit And Component Test, Agile Software Development.*

TABLA DE CONTENIDO

Pág.

LISTA DE FIGURAS

LISTA DE GRÁFICAS

LISTA DE TABLAS

GLOSARIO

CALIDAD DE SOFTWARE: según [1] se refiere a la concordancia con los requerimientos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo documentados y con las características implícitas que se espera de todo software elaborado de forma profesional.

CARACTERIZAR: En [2] es una descripción u ordenamiento conceptual que se hace desde la perspectiva de la persona que la realiza. Esta actividad de caracterizar (que puede ser una primera fase en la sistematización de experiencias) parte de un trabajo de indagación documental del pasado y del presente de un fenómeno, y en lo posible está exenta de interpretaciones, pues su fin es esencialmente descriptivo.

DESARROLLO ÁGIL DE SOFTWARE: Como se menciona en [3] es un concepto usado en el desarrollo de software para describir las metodologías de desarrollo incrementales.

HEURÍSTICAS: técnica de la indagación y del descubrimiento.

ITERACIÓN: en Ingeniería de Software es la práctica de desarrollar y entregar componentes que van creciendo en funcionalidad, comúnmente asociado al desarrollo ágil.

PROCESO DE EVALUACIÓN: tiene como objetivo proveer una definición detallada de la evaluación, para guiar al evaluador de la forma más clara posible.

PRODUCTO SOFTWARE: para considerar al software como un producto, debe existir documentación completa y actualizada que describa los programas, los datos, la forma de uso, entre otros. Está compuesto de: código fuente del programa, código objeto y documentación.

PRUEBAS AUTOMÁTICAS: Según [4] consisten en utilizar un software o herramienta para controlar la ejecución de las pruebas y la comparación de los resultados reales con los esperados.

PRUEBAS DE COMPONENTES: según [5] son aquellas que tienen como objetivo localizar defectos y comprobar el funcionamiento de los módulos del software, programas, objetos, clases entre otros; y pueden probarse independientemente del sistema al que pertenece.

PRUEBAS DE SOFTWARE: consiste en la verificación del comportamiento y calidad de un software con base en un conjunto definido de pruebas debidamente seleccionadas.

PRUEBAS DE UNIDAD: según [6] son aquellas que verifican el funcionamiento aislado de las partes de un software y se pueden probar independientemente. De acuerdo con el contexto, las partes podían ser subprogramas individuales o un componente más grande formado por unidades estrechamente relacionadas.

PRUEBAS MANUALES: Según [4] requieren en todo el proceso de validación que un probador interactúe con cada caso de prueba para analizarlo y reportar los resultados.

INTRODUCCIÓN

Desde hace varias décadas, la Ingeniería de Software ha permitido que las compañías mejoren su productividad, eficiencia y competitividad, gracias a que aporta soluciones que fortalecen los diferentes procesos de los sistemas empresariales, esto ha ocasionado que el interés por adquirir uno de estos productos crezca, y con ello que el mercado de producción de software aumente y sea más exigente al momento de ofrecer uno de estos productos; por esta razón, es esencial asegurar la calidad de los mismos, ya que será el aspecto que tendrá mayor relevancia, al momento en que un cliente elija entre los diferentes productos de software que pueda tener ofertados. Para asegurar calidad, la Ingeniería de Software ofrece una serie de etapas que permiten conseguirlo; dentro de ellas está la fase de pruebas, la cual tendrá mayor influencia en la calidad de los productos software generados, por ser la etapa encargada de verificar el correcto funcionamiento de los mismos; sin embargo, el proceso de pruebas generalmente es limitado o descartado por el esfuerzo y tiempo que requiere; especialmente cuando se realiza de forma manual; por tal razón, una estrategia que se plantea es la automatización de pruebas, debido a que el consumo de recursos en relación al tiempo de diseño y ejecución de las mismas es menor, en contraste con las pruebas manuales, esto se debe a la implementación de herramientas que facilitan dicha actividad, entre ellas podemos encontrar herramientas de código abierto tales como Selenium, Webdriver, TestNG y SoapUI que facilitan la construcción de pruebas automatizadas, lo que es de gran utilidad en el ámbito empresarial y en otros contextos tales como el académico.

[7] Proponen los cuadrantes de las pruebas ágiles, donde se describe que inicialmente se plantea la realización de pruebas de unidad de componentes, para posteriormente combinar pruebas manuales y automáticas. Con base en lo descrito anteriormente, esta investigación busca caracterizar la forma como se realizan pruebas de unidad y componentes en los proyectos de grado que desarrollan software, en el programa de Ingeniería de Sistemas de la Universidad de Nariño. De acuerdo con la caracterización, se diseñará un proceso para el desarrollo y ejecución de este tipo de pruebas. Finalmente se validará el proceso diseñado en un ambiente experimental.

La investigación busca mediante la recopilación y análisis de información de las pruebas utilizadas en los proyectos de grado que involucran software, en el programa de Ingeniería Sistemas de la Universidad de Nariño, plantear un proceso para el desarrollo y ejecución de pruebas automáticas de unidad y componentes, que permita transitar de las pruebas manuales a pruebas automáticas desde el desarrollo ágil de software.

I. CONTEXTUALIZACIÓN

A. LÍNEA DE INVESTIGACIÓN

La investigación pertenece a la línea: SOFTWARE Y MANEJO DE INFORMACIÓN, aprobada según el Acuerdo No 045 de octubre de 2002, ya que se encuentran inmersos tanto el análisis y diseño de sistemas como la ingeniería de software, considerando que tiene como enfoque principal el desarrollo de pruebas automáticas para la evaluación de software.

B. PLANTEAMIENTO DEL PROBLEMA

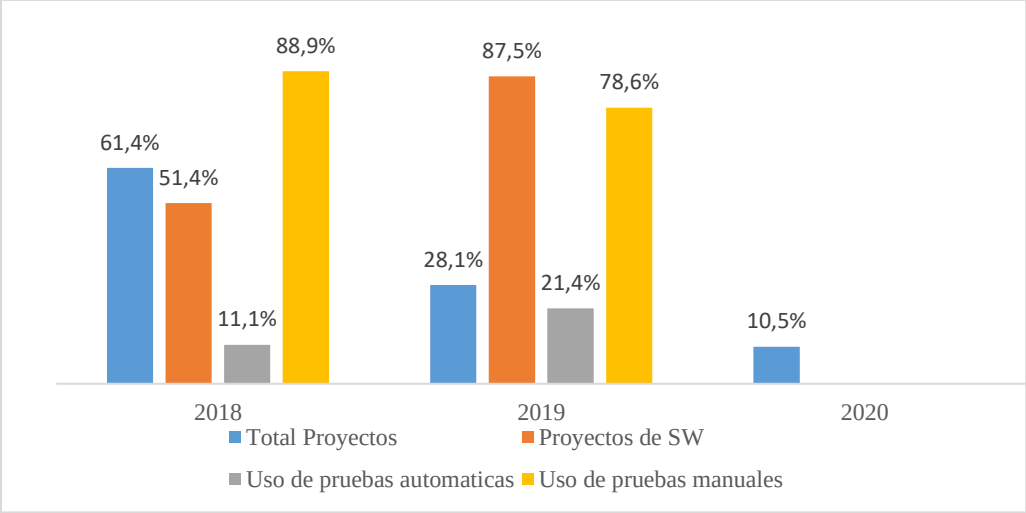
Con el incremento de la complejidad para desarrollar software, cada vez cobra más relevancia el asegurar la calidad de los productos a través del desarrollo de pruebas. Existen múltiples factores que inciden en el nivel de complejidad al realizar pruebas de software, tales como las entradas de usuario, dispositivos/pantallas, ambientes heterogéneos, condiciones de ejecución, contexto, complejidad de dominio, los diferentes ambientes de despliegue, restricciones de aplicación del software y presión continua para entregas frecuentes, ya que permiten evidenciar fallos en los sistemas [8]. Por ejemplo, si se quiere desarrollar una aplicación para un dispositivo móvil, los casos de prueba que se presentan debido al número de modelos, versiones y tamaños en el mercado, corresponden a combinaciones entre estos elementos, tal es el caso de la línea de teléfonos inteligentes marca iPhone que tiene vigentes en el año 2021 diez modelos, cada uno posee de una a cuatro versiones. Además, por cada modelo existen tres diferentes tamaños de dispositivo [9]. Realizando la combinación de modelos, versión y tamaño, se obtiene un total de 21 dispositivos en los cuales se tienen que realizar pruebas del producto software. El esfuerzo de aplicar pruebas manuales para los casos mencionados representaría un incremento considerable en los costos, causando que el presupuesto destinado al desarrollo se invierta solo en esta fase, y que contradictoriamente dentro de los proyectos siempre es el más bajo.

El desarrollo de software es una de las modalidades que con frecuencia seleccionan los estudiantes del programa de Ingeniería de Sistemas para obtener su título profesional. Según el trabajo realizado por [10] entre los años 2010 y 2017, el porcentaje de trabajos de grado donde se realiza productos software corresponde al 54,5% (aproximadamente 114). Ahora bien, para desarrollar un producto software de alta calidad que reciba el visto bueno por parte de los jurados y tenga acogida en la comunidad, se debe seguir unas etapas o procesos de desarrollo que aseguren que el software cumple tanto con los requisitos de funcionalidad y no funcionalidad definidos. En este sentido, la realización de pruebas de software cobra gran relevancia. Sin embargo, según [11] esta etapa suele estar sujeta a la presencia de inconvenientes asociados con la ejecución de pruebas manuales, tales como identificación tardía de errores en los resultados obtenidos de la evaluación, mayor consumo de recursos, utilización de personas de otras áreas en funciones de la fase de pruebas y el desarrollo de requerimientos con plazos limitados.

Lo anterior viene sucediendo por el desconocimiento de incluir pruebas automatizadas para realizar esta tarea. [12] manifiesta que actualmente, gran parte del esfuerzo para probar un proyecto de construcción de software está apoyado por herramientas. Además, plantea que la prueba manual es laboriosa y propensa a errores, y no soporta el mismo tipo de controles de calidad que son posibles con las pruebas automatizadas.

Al realizar una exploración de los trabajos de grado que desarrollaron software entre el 2018 y 2020 (Ver Gráfica 1) se encontró un total de 57 proyectos de grado del Programa de Ingeniería de Sistemas de la Universidad de Nariño, suministrados por el Departamento de Sistemas, de los cuales aproximadamente el 56,1% (equivalente a 32 proyectos) de los trabajos de grado desarrollaron o mejoraron un software, entre estos el 93,8% (equivalente a 30 proyectos) documentan la fase de pruebas, encontrando que solo el 15,6% (equivalente a cinco proyectos) efectuaron pruebas automatizadas mediante el uso de herramientas, las cuales fueron:

- JMeter para evaluar pruebas de estrés, simulando la carga de múltiples usuarios.
- Laravel el cual incluye un componente que facilita el proceso de pruebas unitarias con PhpUnit, utilizando ficheros de pruebas automatizadas, elaboradas en el proyecto.
- Examinaitor para comprobar si el sistema cumple con los lineamientos de accesibilidad.
- Eclipse Neón con la clase AllTest.java, es una suite de pruebas que ejecuta los paquetes unit, integration e Interfaz de Usuario (UI por sus siglas en inglés), que contienen las clases para la ejecución de pruebas unitarias, de integración y pruebas de interfaz gráfica respectivamente.



Gráfica 1. Proyectos de grado del Programa de Ingeniería de Sistemas de la Universidad de Nariño (2018 - 2020)

Fuente: esta investigación.

Al indagar a un grupo de cinco egresados por las razones que tuvieron para no implementar pruebas automáticas en los proyectos de grado, cuya fase de validación de software se efectuó de forma manual, se encontró que fundamentan esta etapa en conocimientos adquiridos durante su carrera, expresando desconocimiento de aplicación de pruebas de tipo automático. Además, se logró evidenciar el uso de recursos proporcionados por el Departamento de Informática, tales como formatos para la aplicación de pruebas manuales, lo que pudo haber influido en la elección del tipo de prueba.

Por otra parte, [7] proponen los cuadrantes de pruebas ágiles como se puede observar en la Fig. 1. El primer paso para la automatización de pruebas se encuentra en el cuadrante Q1, donde se plantea que se deben realizar las pruebas de unidad y las pruebas de componentes; para posteriormente avanzar hacia el cuadrante Q2, donde se pueden combinar pruebas manuales y automáticas. Las pruebas de unidad o unitarias según [6] son aquellas que verifican el funcionamiento aislado de las partes de un software y se pueden probar independientemente. De acuerdo con el contexto, las partes podían ser subprogramas individuales o un componente más grande formado por unidades estrechamente relacionadas. Según [5] las pruebas de componentes corresponden con aquellas que tienen como objetivo localizar defectos y comprobar el funcionamiento de módulos software, programas, objetos, clases entre otros; y pueden probarse independientemente del sistema al que pertenece.

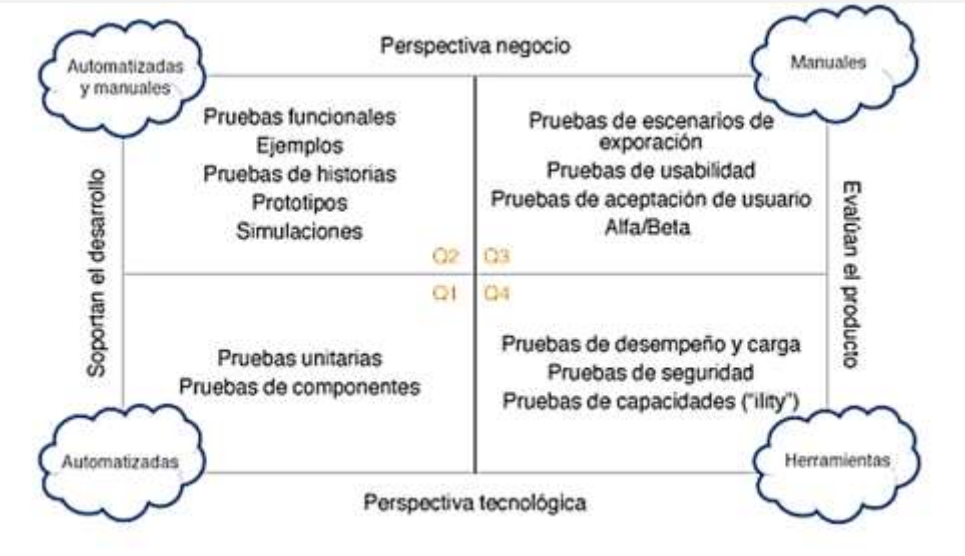


Fig. 1. Cuadrantes de pruebas ágiles.

Fuente: tomada de [7]

Teniendo en cuenta lo anteriormente descrito, el primer paso para automatizar pruebas corresponde con el cuadrante Q1. En este sentido, el trabajo investigativo se centra en buscar una forma para

aportar al proceso de desarrollo de pruebas automáticas de unidad y componentes, en los trabajos de grado que desarrollan software, en el programa de Ingeniería de Sistemas de la Universidad de Nariño desde el desarrollo ágil de software.

C. FORMULACIÓN DEL PROBLEMA

¿Cómo aportar al proceso de desarrollo de pruebas automáticas de unidad y componentes, en los trabajos de grado que desarrollan software, en el programa de Ingeniería de Sistemas de la Universidad de Nariño desde el desarrollo ágil de software?

D. SISTEMATIZACIÓN DEL PROBLEMA

- ¿De qué forma se realizan las pruebas de unidad y componentes en los proyectos de grado que desarrollan software en la Universidad de Nariño?
- ¿De qué manera se puede sistemáticamente desarrollar y ejecutar pruebas automáticas de unidad y componentes desde el desarrollo ágil de software?
- ¿De qué forma se puede dar validez a un proceso de desarrollo y ejecución de pruebas automáticas de unidad y componentes en la Universidad de Nariño?

E. JUSTIFICACIÓN

Según [13], la importancia de las pruebas ha experimentado un crecimiento significativo, gracias a los beneficios para el diseño de software, por lo que el ingeniero de software debe decidir si utilizar herramientas que automáticamente generen pruebas unitarias, o si dicho proceso se efectuará de manera manual, esta decisión puede estar sujeta a factores tales como el grado de complejidad del software, los costos, recursos de tiempo y personal, en este sentido, y compartiendo lo expuesto por [14] referente a que la prueba de software es una tarea crucial en el proceso de desarrollo y que requiere mucho tiempo, en los últimos años se han introducido muchos enfoques y herramientas para la generación automática de las mismas, ya que se ha determinado que aceleran el proceso de evaluación del producto software y la confiabilidad de los resultados obtenidos es mayor, debido a la precisión que estas ofrecen, cabe resaltar que no se debe desacreditar las pruebas manuales, ya que son válidas, útiles y no pueden ser descartadas inmediatamente. Sin embargo, se observa la necesidad de investigar el campo de estudio que abarca la automatización en las pruebas de software, las cuales incluyen planificación, modelación y uso de nuevas herramientas.

Este trabajo investigativo es interesante, porque permitirá conocer la forma como se realizó la fase de pruebas de unidad y componentes, en los proyectos de grado que desarrollaron software en el programa de Ingeniería de Sistemas de la Universidad de Nariño, durante los años 2010 a 2020. Este análisis permitirá efectuar la caracterización de los mismos, obteniendo información acerca

del tipo de pruebas ejecutadas ya sea de forma manual o automática, y en el caso de que se hayan efectuado pruebas automáticas, identificar las herramientas empleadas, con lo que se evidenciará la eficiencia en su aplicación y resultados.

La utilidad de esta investigación se verá reflejada principalmente en el proceso para el desarrollo y ejecución de pruebas automáticas unitarias y de componentes, desde el desarrollo ágil de software, cuyo propósito es ser un referente cuando se quiera implementar este tipo de pruebas en los proyectos de software del programa de Ingeniería de Sistemas de la Universidad de Nariño, ya que al tener un proceso estándar, la complejidad de su aplicación será menor; así como, el tiempo invertido en el testeo del software, razón por la cual los principales beneficiados serán los estudiantes, quienes tendrán a disposición dicho proceso, y los docentes del programa, ya que con base en esta información pueden reorientar la metodología de formación en relación con la fase de pruebas. Otro aporte significativo, será la validación del proceso mencionado en un ambiente experimental, debido a que permitirá comprobar su eficiencia en relación con los diversos requisitos que puede tener un software, generando confiabilidad y seguridad tanto en la aplicación del proceso como en los resultados obtenidos. Al llevar a cabo esta investigación se identificarán oportunidades de mejora en cuanto a la evaluación de pruebas de unidad y componentes aplicadas automáticamente, dado que el desarrollo de software es una de las alternativas más común en los proyectos de grado, y efectuar la fase de validación es imprescindible.

Lo novedoso de esta investigación está dado por el desarrollo de un proceso de pruebas automáticas de unidad y componentes, ya que se aborda el tema de automatización de pruebas desde un enfoque de desarrollo ágil de software, siendo este un estudio del cual no se ha encontrado referencias en la Universidad de Nariño, a pesar de ser algo que se ha implementado con mayor frecuencia en los *Frameworks* de programación como Laravel, Eclipse, entre otros, y herramientas que se especializan en este campo, como es el caso de Selenium. Además, es importante destacar que, en el ámbito laboral la implementación de pruebas automáticas es un tema común.

F. OBJETIVOS

1) Objetivo general

Aportar al proceso de desarrollo de pruebas automáticas de unidad y componentes, en los trabajos de grado que desarrollan software, en el programa de Ingeniería de Sistemas de la Universidad de Nariño desde el desarrollo ágil de software.

2) Objetivos específicos

Caracterizar la forma como se realizan las pruebas de unidad y componentes en los proyectos de grado que desarrollan software en la Universidad de Nariño entre los años 2010 a 2020.

Diseñar un proceso para el desarrollo y ejecución de pruebas automáticas de unidad y componentes desde el desarrollo ágil de software, con base en la caracterización.

Validar el proceso para desarrollo y ejecución de pruebas automáticas de unidad y componentes diseñado, en un ambiente experimental.

II. MARCO TEÓRICO

A. MARCO DE ANTECEDENTES

Los autores [15] proponen un modelo de simulación que utiliza la técnica de modelado de dinámica del sistema, el cual ayuda en la toma de decisiones sobre si automatizar sus procesos de prueba, y en caso de hacerlo en qué medida efectuarlo, realizaron el estudio en colaboración con una empresa de software de Canadá en la cual compararon pruebas totalmente manuales con pruebas parcialmente automatizadas, lo que proporcionó una comprensión más profunda sobre las fortalezas y debilidades del proceso que llevaban a cabo. Este antecedente, sin bien realiza un análisis en profundidad sobre las pruebas que se llevan a cabo al construir software, no se observa que propongan el diseño de un proceso para el desarrollo y ejecución de las mismas; simplemente se limitan a comparar el impacto entre pruebas manuales y automáticas.

[16] con su estudio analizan las pruebas generadas manualmente en un ambiente de desarrollo donde se presentan cambios en el código, se analizan los datos de la selección de pruebas manuales y recopila información en tiempo real de 14 desarrolladores en un total de 450 sesiones, posteriormente se hace una comparación con la selección de pruebas de forma automática. Como resultado al elegir las pruebas de las 450 sesiones, la selección manual escogió una cantidad mayor en comparación a la toma automática, teniendo un porcentaje de uso en tiempo del 73,0% y 27,0% respectivamente, mostrando mayor pérdida de tiempo en la selección manual respecto a la selección de pruebas automáticas, aunque con errores que esta minoría de pruebas no pudo identificar, mostrando la necesidad de mejores técnicas de selección de pruebas automáticas que se puedan integrar en un entorno para desarrolladores. La diferencia entre este estudio y la investigación planteada, radica en el ambiente en dónde se desarrolla la recolección para el análisis de pruebas automáticas y manuales, dado que es un ambiente de desarrollo en donde se presentan cambios en el código, contrario a este estudio, dónde la información se extraerá de proyectos de grado que involucran software en el programa de Sistemas de la Universidad de Nariño, cuya fase de codificación ha sido finalizada.

Los autores [14]en su estudio exploratorio comparan la legibilidad de casos de prueba escritos manualmente con las clases que evalúan. Además, se profundiza dicho análisis observando la legibilidad de los casos de prueba generados automáticamente. Mostrando resultados que sugieren que los desarrolladores tienden a descuidar la legibilidad de los casos de prueba, y aquellos generados automáticamente son con frecuencia menos legibles que los escritos manualmente. Esta exploración se limita a medir la legibilidad de las pruebas tanto automáticas como manuales, sin proponer una caracterización de los datos recolectados, ni diseñar un proceso basado en dicha caracterización para pruebas automáticas de unidad y componentes.

En la Universidad Técnica Federico Santa María [17] realizó un estudio acerca de las herramientas de automatización para la realización de pruebas de software, con el objetivo de efectuar un análisis evaluativo y comparativo de las diferencias de esfuerzo entre la ejecución de pruebas manuales

contra las pruebas automatizadas, obteniendo como resultado tiempos de ejecución hasta 20 veces más rápidos con las pruebas automatizadas, determinando también, el número de iteraciones en que la prueba automática es más eficiente que la prueba manual. Dicho estudio permite dimensionar la efectividad y disminución del trabajo obtenido al automatizar las pruebas, limitándose a identificar las herramientas requeridas para ello, mientras que esta investigación realiza la exploración de proyectos de grado que pueden incluir pruebas manuales y el uso de herramientas para pruebas automáticas, con lo que se efectuará una caracterización para el diseño de un proceso.

[18] realizó un estudio que se dio lugar en Perú, ratificando la confiabilidad generada mediante el uso de herramientas para la automatización de pruebas de software, tales como Specflow y Selenium, las cuales permiten realizar pruebas funcionales. Contar con equipos que se dediquen a realizar estas pruebas de forma paralela al desarrollo, contribuyen a mejorar la calidad del producto software y disminuir los errores de los entregables en los hitos establecidos por los clientes. En comparación, dicho estudio centra su confianza en el uso de herramientas para efectuar la fase de pruebas de forma automática, mientras que para esta investigación se propone un proceso para la automatización de pruebas unitarias y de componentes, cuya confiabilidad se medirá a través de la validación del mismo en un ambiente experimental.

El estudio realizado en la Universidad de Salerno Italia en conjunto con la Universidad de Zurich en Switzerland por [13], se basa en la investigación previa de [19], quién inicialmente compara el comportamiento entre las pruebas automáticas y las generadas manualmente, por lo que en este caso se incluye una revisión de su trabajo utilizando herramientas actuales, complementando su método de investigación al evaluar la capacidad de estas herramientas. Esta investigación permite reconocer el momento propicio para ejecutar pruebas automáticas o manuales al evaluar el software, dado el grado de complejidad y costos que esto implica, por lo cual conocer de la existencia de técnicas y estudios relacionados con este enfoque de automatización en las pruebas, así como la validación de las herramientas que suelen ser usadas, resulta de gran utilidad en la reducción de carga para los desarrolladores, fomentando la calidad del producto software. En este estudio al igual que en la investigación a desarrollar, se indaga sobre las pruebas para recolectar información, pero con el objetivo de analizar el comportamiento y elegir el mejor momento de aplicación de alguno de los dos tipos de pruebas, mientras que la recolección de información de esta investigación servirá para generar una caracterización en la que se fundamentará un proceso de automatización de pruebas de unidad y componentes en el Programa de Ingeniería de Sistemas de la Universidad de Nariño, donde no se han registrado este tipo de pruebas.

La investigación avalada por varias e importantes instituciones en Beijin China, realizada por [20] enuncia en primera instancia la importancia que cobra la prueba de software en el ciclo de vida del desarrollo del mismo, y como las pruebas automáticas pueden reducir el costo de las pruebas manuales aumentando la confiabilidad de las mismas. El estudio propone un modelo de arquitectura de pruebas automáticas de software, complementando estudios previos basados principalmente en el desarrollo de técnicas que buscan incrementar la calidad del producto

software. La relevancia que tiene el modelado de las pruebas, radica en que una vez se tiene el previo diseño de la misma, la complejidad en su ejecución disminuye. La investigación propone un proceso que tiene como objetivo incrementar el uso de pruebas automáticas basándose en la arquitectura de las mismas para la generación de un modelo, en tanto el proceso que se diseñará e implementará en esta investigación se fundamentará en la caracterización de la forma como se realizan las pruebas automáticas de unidad y componentes.

[21] desarrollaron un estudio de investigación que se dio lugar en la Corporación Universitaria Remington e Institución Universitaria de Envigado, con el cual se propone un modelo para determinar la madurez de la automatización de las pruebas como área de investigación y desarrollo en la industria del software, describiendo el proceso de *testing* automático, analizando su eficacia y eficiencia, llegando a la conclusión de que el nivel de madurez actual de la automatización de pruebas del software es "adolescente", este estudio concibe la automatización de pruebas a través de la investigación, en lugar de solo apuntar a la mejora de la producción y el rendimiento de los equipos. El estudio propone un modelo que tiene como objetivo describir el proceso de pruebas automáticas para la realización de una investigación en un contexto donde fueron implementadas previamente, a diferencia del proceso a desarrollar en esta investigación cuya aplicación se llevará cabo en un entorno universitario.

De acuerdo con [22] la representación de un proceso puede efectuarse mediante diversas formas de modelado, que se clasifican según la descripción o detalles que presentan; entre ellas están: mapas de procesos, los cuales son diagramas sencillos de secuencias de actividades; descripción de procesos, que muestra mayor información de un proceso, y modelos de procesos, que son gráficos con un nivel de detalle superior, permitiendo analizar y simular el proceso.

Según [22] los principales elementos que BPMN utiliza para representar un proceso son: actividades, eventos, *Gateways* y flujos de secuencia.

Las actividades son las acciones realizadas en un proceso de negocio, pueden clasificarse acorde con el nivel de detalle que representan, estas pueden ser: atómicas y compuestas. (Ver Fig. 2).

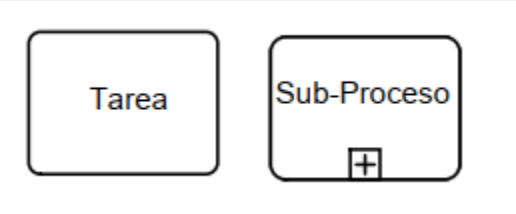


Fig. 2 Representación de actividad atómica y compuesta.

Fuente: tomada de [22]

Los eventos son sucesos que pueden ocurrir dentro del flujo de un proceso de negocio, iniciándolo, retrasándolo, interrumpiéndolo o finalizándolo; generalmente tienen un resultado o un disparador que lo inicia. Se representan mediante un círculo, como se muestra en la Fig. 3, el cual puede variar según la acción que desempaña en el flujo de procesos.

Fig. 3. Evento de inicio y fin.

Fuente: tomada de [22]

Las *Gateways* son elementos que controlan la división o unificación de los caminos, que se presentan en el flujo del proceso, se representan mediante un rombo, el cual puede variar según la acción que desempaña (Ver Fig. 4).

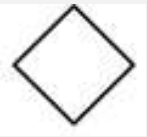


Fig. 4. Gateway exclusivo.

Fuente: tomada de [22]

El flujo de secuencia es el encargado de conectar los elementos que participan en la representación del proceso de negocio, mostrando los caminos que se pueden tomar, su forma es una flecha como se observa en la Fig. 5.

Fig. 5. Flujo de secuencia.

Fuente: tomada de [22]

El modelado de un proceso de negocio puede realizarse mediante el uso de herramientas que facilitan su representación, entre ellas están: Bizagi Modeler[1], Flokzu[2], BPMN.IO[3], Lucichart[4], entre otras.

[1] Bizagi Modeler es una marca registrada de la empresa Bizagi que se puede descargar de https://www.bizagi.com/es/plataforma/modeler

[2] Flutzo es una marca registrada de Flutzo, se puede acceder a la herramienta desde https://www.flokzu.com/ES

[3] BPMN.IO es una marca registrada de Camunda Services GmbH, se puede acceder a la herramienta desde https://bpmn.io

[4] Lucichart es una marca registrada de la empresa Lucid Software Inc que se pude acceder desde https://www.lucidchart.com/pages/es

III. METODOLOGÍA

Esta investigación es de corte cuantitativo, porque según [23] representa un conjunto de procesos secuenciales y probatorios, por lo tanto, no se pueden saltar los pasos debido a que el orden es riguroso que parte de una idea que una vez delimitada, se derivan en objetivos y preguntas de investigación, de las preguntas se establecen hipótesis y se determinan variables y la validez de los resultados, se soportará a través de un proceso estadístico descriptivo debido a que se realizan mediciones, se utilizan técnicas de recolección y se analizan los datos procediendo de manera inductiva donde se obtendrán conclusiones empíricas.

El enfoque para esta investigación es empírico-analítico [24], debido a que se basa en las experiencias propias para plantear un tránsito de las pruebas manuales a las pruebas automáticas en el programa de Ingeniería de Sistemas de la Universidad de Nariño, pero desde la perspectiva del desarrollo ágil de software.

El tipo de investigación es descriptiva [24], porque se pretende recolectar datos sobre la forma como se realizan las pruebas de unidad y componentes, en los proyectos de grado que desarrollan software en la Universidad de Nariño entre los años 2010 a 2020, y de esta manera describir las propiedades que tiene esta etapa en el proceso de desarrollo de software, mediante la caracterización de la información obtenida de estos documentos. Además, la investigación es propositiva [24], porque se diseñará y planteará un proceso de apoyo para el desarrollo y ejecución de pruebas automáticas de unidad y componentes desde el desarrollo ágil de software.

El proceso de desarrollo de esta investigación incluye tres etapas. En la TABLA I, se muestran las etapas y actividades integradas en la metodología de la investigación.

TABLA I. PROCESO DE DESARROLLO DE LA INVESTIGACIÓN

Etapa 1	*Caracterizar pruebas de unidad y componente en proyectos de grado de la Universidad de Nariño años 2010 - 2020*
Actividades	1.1. Elaborar un instrumento de recolección de información 1.2. Validar el instrumento de recolección de información 1.3. Aplicar el instrumento de recolección de información 1.4. Analizar los datos recopilados
Etapa 2	*Diseñar un proceso para el desarrollo y ejecución de pruebas de automáticas unitarias desde el desarrollo ágil de software*
Actividades	2.1. Definir actores del proceso 2.2. Establecer las fases y actividades del proceso 2.3. Definir los flujos de control por fase del proceso 2.4. Elaborar el modelo del proceso utilizando una notación
Etapa 3	*Validar el proceso para desarrollo y ejecución de pruebas unitarias y de componentes diseñado, en un ambiente experimental con proyectos activos de los semestres A y B del año 2021*
Actividades	3.1. Seleccionar la población participante 3.2. Planear un taller de capacitación para el desarrollo del proceso de desarrollo y ejecución de pruebas automáticas 3.3. Organizar el taller de capacitación 3.4. Desarrollar el taller de capacitación 3.5. Evaluar las percepciones de los participantes en relación con el taller

Fuente: esta investigación.

IV. RESULTADOS DE LA INVESTIGACIÓN

Una vez finalizada la investigación se espera obtener una caracterización con la forma como se realizan las pruebas de unidad y automáticas en los proyectos de grado que desarrollan software en el programa de Ingeniería de Sistemas de la Universidad de Nariño. La caracterización permitirá representar el proceso de desarrollo y ejecución de pruebas de unidad y componentes mediante un modelo de proceso de negocio, bajo notación BPMN con actores, actividades, canales y flujos de control. Además, el proceso se someterá a una validación en un ambiente experimental.

A. CARACTERIZAR LA FORMA COMO SE REALIZAN LAS PRUEBAS DE UNIDAD Y COMPONENTES

En esta sección se presenta los resultados de la caracterización de la forma como se realizan las pruebas de unidad y componentes en los proyectos de grado que desarrollan software en la Universidad de Nariño entre los años 2010 a 2020. Para este proceso (Ver Fig. 6), se realizaron las siguientes actividades.

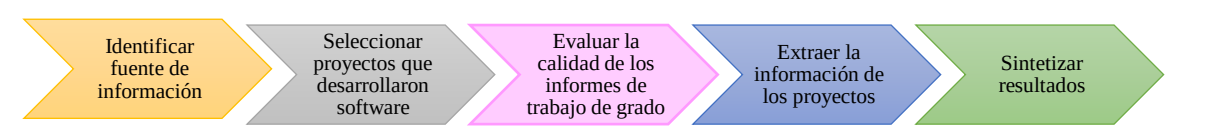


Fig. 6. Proceso de caracterización.

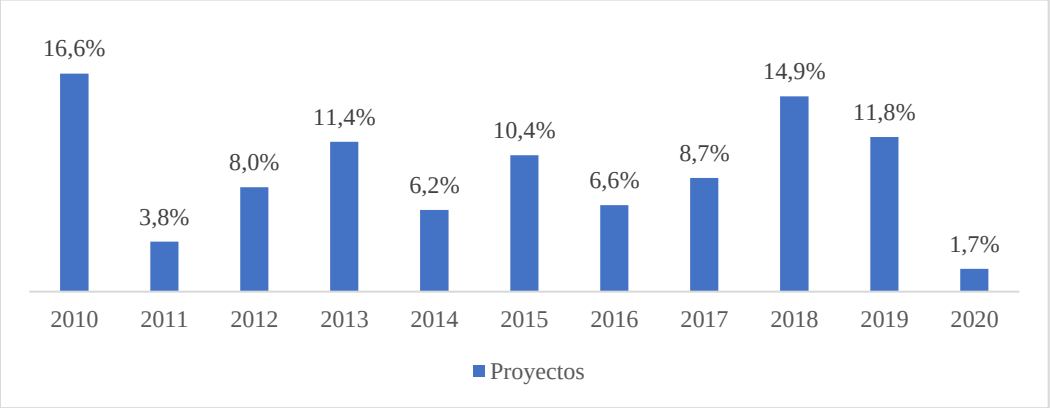
Fuente: esta investigación

1) Identificar fuente de información

El primer paso para identificar las fuentes de información fue establecer un diálogo con el director del Departamento de Sistemas, el Mg Luis Obeimar Estrada Sapuyes, quién suministró una relación con la información correspondiente a los proyectos de grado que desarrollaron software del Programa de Ingeniería de Sistemas entre los años 2010 al 2020 (Ver Anexo A). Según aclaraciones brindadas por el director del Departamento, éste manifestó que las principales fuentes de información para la revisión de los documentos físicos son: el Repositorio Virtual de la Biblioteca de la Universidad de Nariño (SIRED), donde se encuentra el Sistema Institucional de Recursos Digitales disponible en el portal http://sired.udenar.edu.co/cgi/search/advanced; y el Repositorio de la secretaría del Departamento de Sistemas, el cual contiene los documentos físicos almacenados en CDs o DVDs.

Se encontraron un total de 289 proyectos de grado desarrollados por 442 estudiantes (Ver Anexo A). El año en el cual se desarrolló la mayor cantidad de proyectos de grado fue el 2010 con un

16,6% (equivalente a 48 proyectos), en los periodos de 2011 al 2013 y 2016 al 2018, se presenta un incremento en el desarrollo de proyectos de grado, que van de 11 proyectos equivalentes al 3,8% hasta los 33 proyectos equivalentes al 11,4% y de 19 proyectos que corresponden al 6,6% hasta 43 proyectos que pertenecen al 14,9% respectivamente; mientras que el año 2020 presenta una frecuencia del 1,7% (equivalente a cinco proyectos) siendo el año con menor cantidad de proyectos de grado realizados (Ver Gráfica 2).

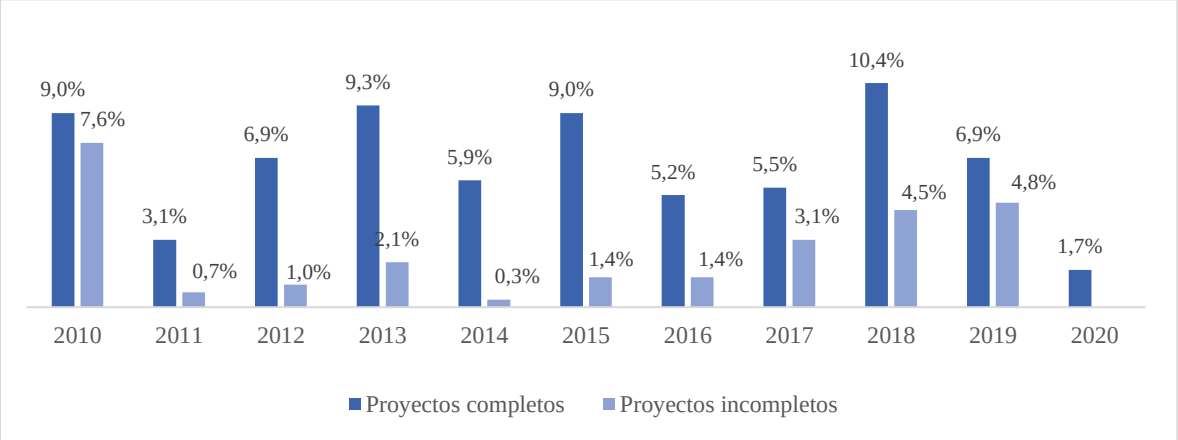


Gráfica 2. Proyectos de grado por año.

Fuente: esa investigación.

De estos documentos, se buscó información correspondiente al año de culminación según el acuerdo de finalización. Los proyectos que no tenían dicho dato fueron consultados en el repositorio SIRED. Con la información obtenida, se clasificó al 73,0% (equivalente a 211 proyectos) como completos, ya que cuentan con el año de finalización; mientras que el 27,0% (equivalente a 78 proyectos) fueron clasificados como incompletos, debido a que tanto en el Anexo A como en el repositorio no fue posible encontrar su año de finalización.

De acuerdo con lo anterior, se infiere que, al mes de mayo de 2021, es muy probable que los proyectos incompletos aún no hayan finalizado. Al observar el comportamiento de los proyectos incompletos (Ver Gráfica 3), se presenta un incremento en los años 2011 a 2013 y de 2014 a 2019, partiendo de un 0,7% hasta un 2,1% y de 0,3% a un 4,8% respectivamente. Sería importante analizar las razones por las que se viene presentado un incremento en los proyectos que aún no finalizan.



Gráfica 3. Proyectos de grado completos e incompletos según año de finalización.

Fuente: esta investigación.

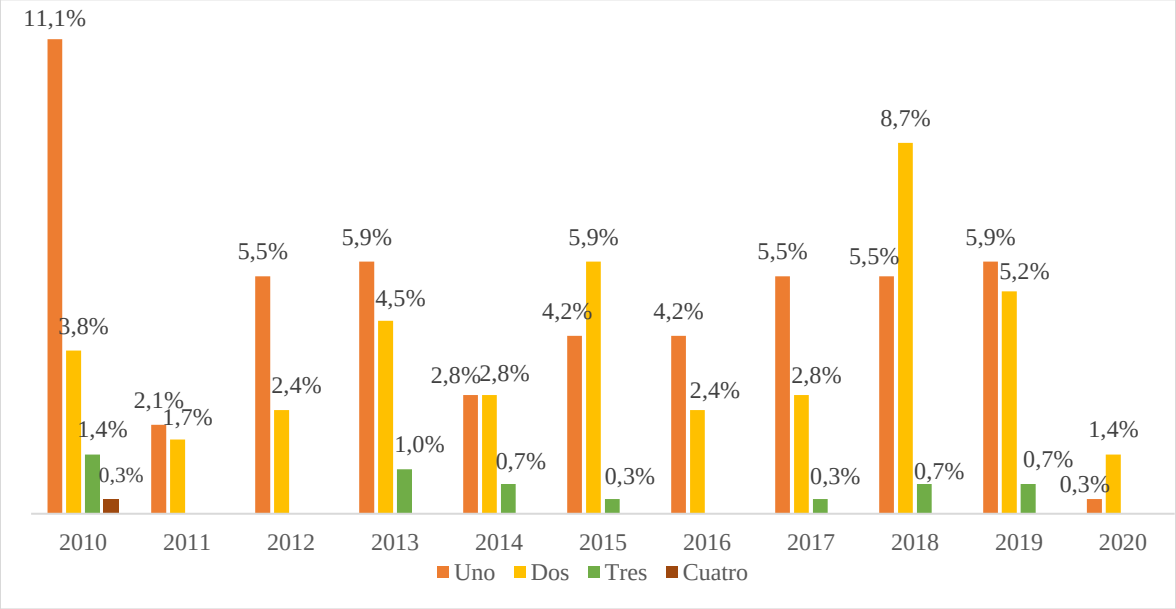
El número de integrantes que conforman los proyectos de grado entre los años 2010 y 2020 (Ver Gráfica 4), indican que hasta el año 2013 la opción de un autor es la que predomina, categoría que lidera en el año 2010; con una frecuencia del 11,1% (equivalente a 32 autores que trabajaron individualmente).

En el año 2018 se presenta mayor tendencia por la categoría de dos autores, con un 8,7% (equivalente a 25 proyectos).

El año 2010, es el único en el que se identifican trabajos elaborados por cuatro autores, con una frecuencia del 0,3% (equivalente a un proyecto).

Los proyectos realizados por tres autores muestran una frecuencia que oscila entre el 0,3% y 1,4%.

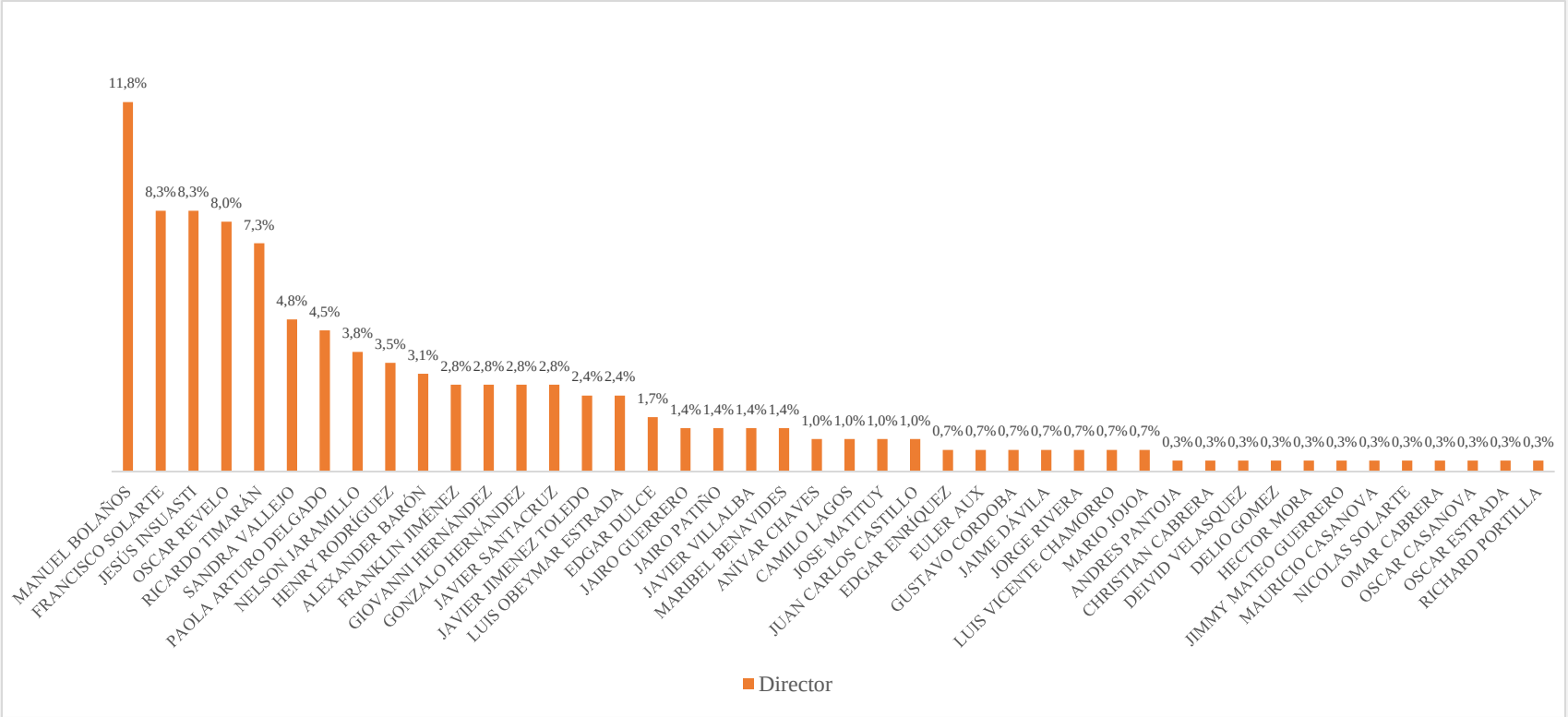
En general, los porcentajes de la conformación de integrantes en los proyectos de grado, está dado por: 52,9% para un autor (equivalente a 153 proyectos), 41,5% (equivalente a 120 proyectos) para dos autores, 5,2% (equivalente a 15 proyectos) para tres autores y un 0,3% (equivalente a un proyecto) para la categoría de cuatro autores.



Gráfica 4. Número de autores por proyecto de grado.

Fuente: esta investigación.

En los trabajos de grado realizados entre los años 2010 a 2020, se obtuvo un total de 45 docentes que se encargaron de asesorar proyectos de grado como directores. En la Gráfica 5, se observa que el docente con mayor cantidad de trabajos de grado dirigidos es el Ingeniero Manuel Bolaños, con una frecuencia del 11,8% que corresponde a 34 proyectos, mientras que la menor frecuencia es del 0,3% que pertenece a un proyecto, siendo esta la que más se repite.

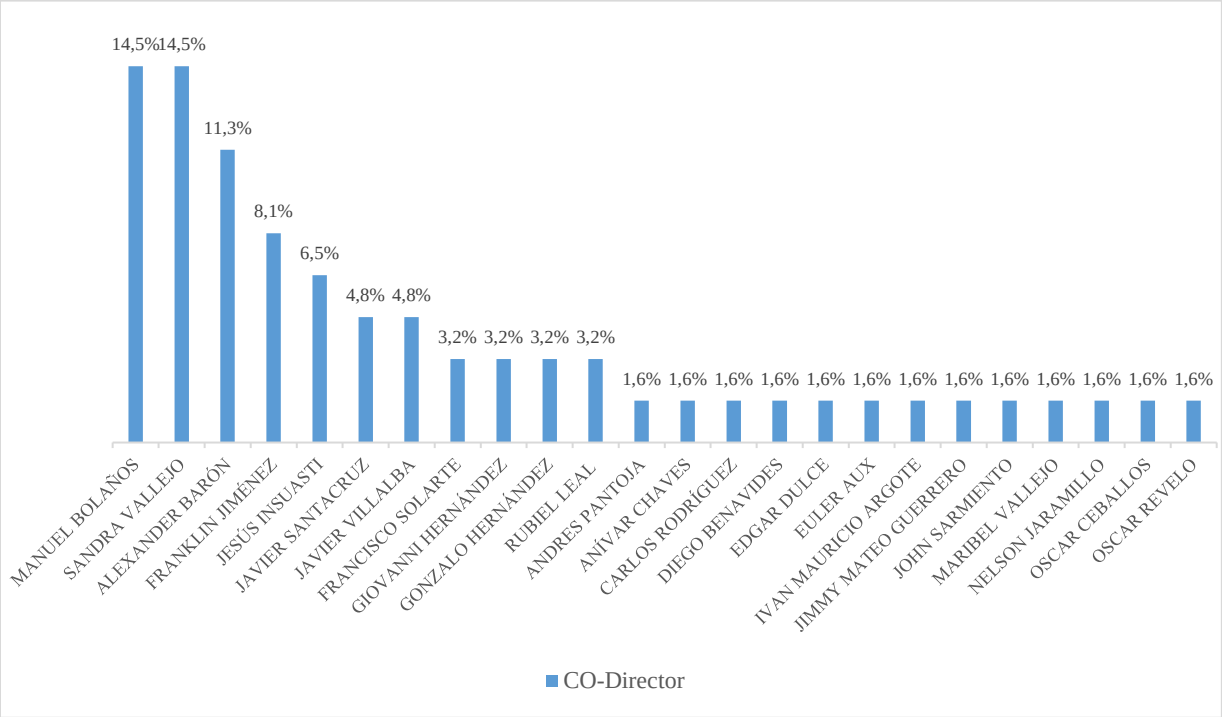


Gráfica 5. Proyectos de grado por director.

Fuente: esta investigación.

Por otra parte, en aquellos trabajos de grado en que interviene un co-director, la mayor frecuencia observada es de nueve proyectos, equivalente al 14,5%, cifra alcanzada por los docentes Sandra Vallejo y Manuel Bolaños; en comparación a la menor frecuencia correspondiente un proyecto, la cual es la que más repite dentro del grupo de 25 docentes, quienes apoyaron los trabajos de grado como co-directores. Esta información se muestra en la Gráfica 6.

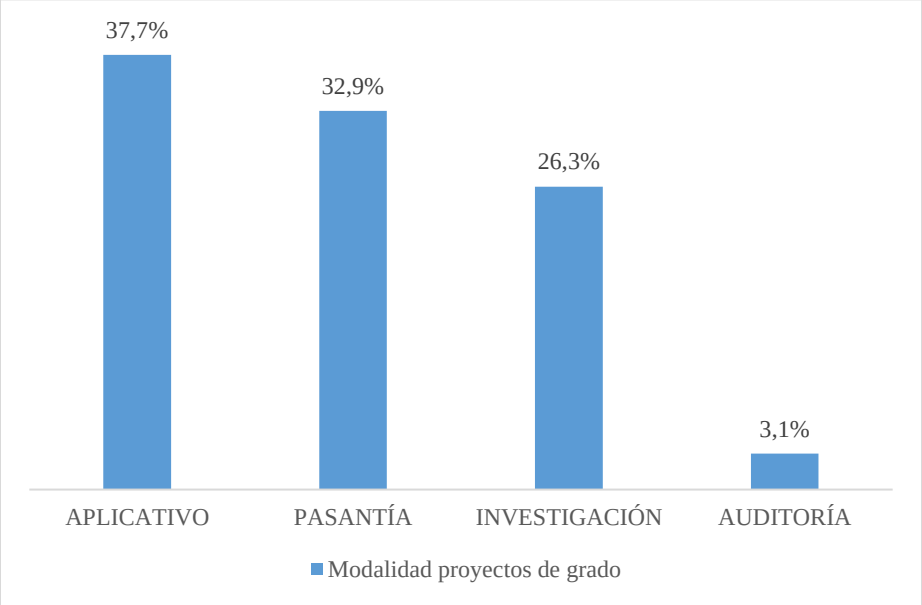
En relación con lo anterior, se pudo observar que el docente que más se involucró en la asesoría para la realización de proyectos de grado, como Director y Co-Director fue el Ingeniero Manuel Bolaños liderando las frecuencias de los análisis previamente realizados.



Gráfica 6. Proyectos de grado por Co-director

Fuente: esta investigación.

Categorizando los proyectos de grado de acuerdo con su modalidad (Ver Gráfica 7), se tiene que la de preferencia entre los años 2010 a 2020 es la modalidad de aplicativo con un 37,7% (equivalente a 109 proyectos), en comparación con la modalidad de auditoría con 3,1% (equivalente a nueve proyectos) cuya frecuencia es la menor.



Gráfica 7. Modalidad de proyectos de grado.

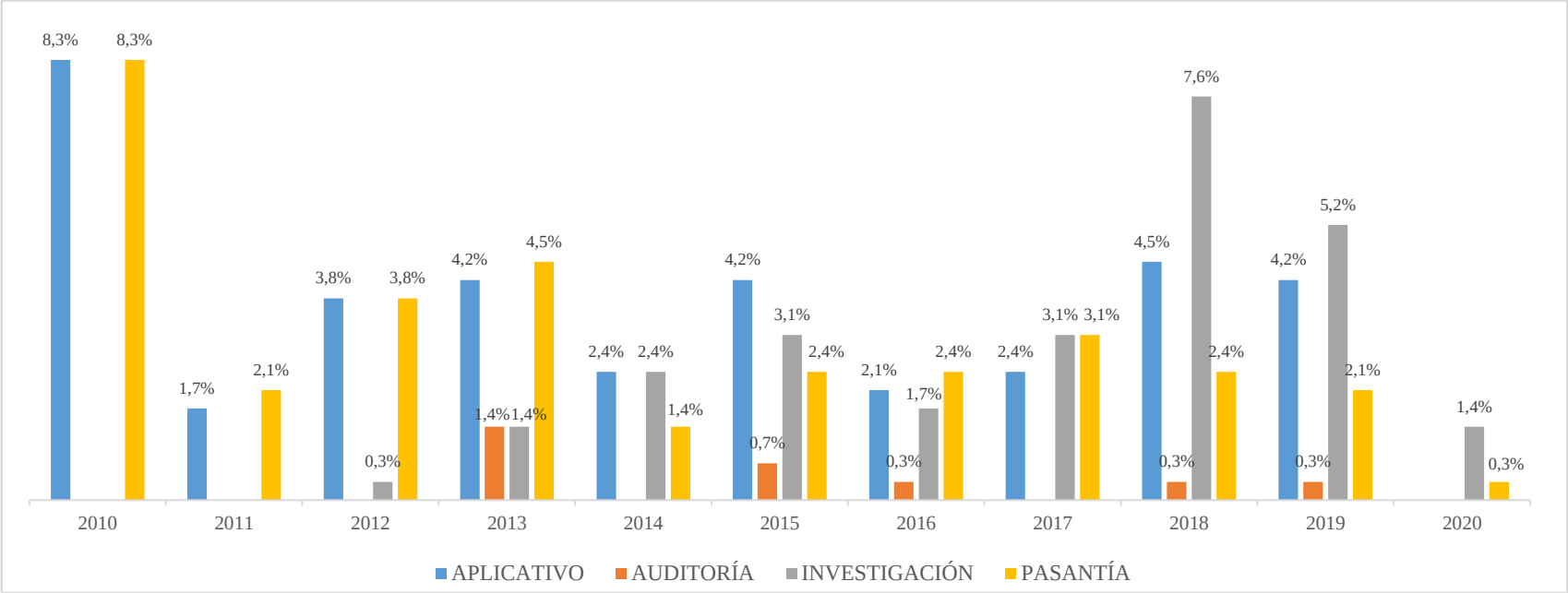
Fuente: esta investigación.

Clasificando los proyectos de grado por modalidad y distribuyéndolos según su año de finalización, como se observa en la Gráfica 8, las modalidades que mayor frecuencia presentan en un año, son la de aplicativo y pasantía, con un 8,3% correspondiente a 24 proyectos en el 2010; además, la modalidad de pasantía es la única existente en todos los años, presenta un aumento que empieza en el 2011 con un 2,1% (equivalente a seis proyectos), terminando en el 2013 con un 4,5% (equivalente a 13 proyectos). Posteriormente, en el año 2014, la cantidad de proyectos de dicha modalidad se mantiene en un rango entre 1,4% (equivalente a cuatro proyectos) y 3,1% (equivalente a nueve proyectos) hasta el 2020.

La modalidad de aplicativo muestra un incremento que pasa de un 1,7% (equivalente a cinco proyectos) a un 4,2% (equivalente a 12 proyectos) entre los años 2011 y 2013. Además, entre los años 2016 y 2018, aumenta del 2,1% (equivalente a seis proyectos) a un 4,5% (equivalente a 13 proyectos).

Los proyectos de investigación surgen a partir del año 2012 con un 0,3% (equivalente a un proyecto), presentando un incremento en la cantidad de proyectos, respecto a las otras modalidades en cada uno de los años trabajados; llegando incluso a liderar en los tres últimos años (2018, 2019 y 2020), denotando la importancia que ha adquirido como enfoque para desarrollo de proyectos de grado con el transcurrir los años.

La modalidad de auditoría es la de menor frecuencia, al no superar el 1,7% (equivalente a cuatro proyectos), mostrando un comportamiento que tiende a desaparecer por su escaso desarrollo durante los años del periodo trabajado.



Gráfica 8. Modalidad de los proyectos de grado por año.

Fuente: esta investigación.

2) Seleccionar proyectos que desarrollaron software

Esta actividad tiene como propósito identificar los proyectos de grado que entre los años 2010 y 2020 desarrollaron productos software. Para la selección, se definieron unos criterios de inclusión y exclusión (Ver TABLA II) de acuerdo con una seria de filtros (Ver TABLA III).

TABLA II. CRITERIOS DE INCLUSIÓN Y EXCLUSIÓN DE LOS FILTROS.

Criterio de inclusión	Criterio de exclusión
a. El proyecto tiene una fecha de finalización.	a. No posee fecha de finalización
b. La fecha de finalización del proyecto corresponde con un año ubicado entre el 2010 y el 2020	b. La fecha de finalización del proyecto es menor al año 2010
c. El proyecto tiene un informe final.	c. No es posible ubicar un informe final de investigación
d. En el proyecto se desarrolló un producto software	d. El proyecto no tiene dentro de sus entregables un producto software

Fuente: esta investigación

TABLA III. DESCRIPCIÓN DE FILTROS.

Filtro	Descripción	Criterio aplicado	
		Inclusión	Exclusión
F1	Buscar los proyectos que tienen fecha de finalización	a	a
F2	Seleccionar los proyectos con fecha dentro del rango de la investigación	b	b
F3	Buscar los informes finales de los proyectos.	c	c
F4	Leer el título, resumen, problema, objetivos y conclusiones.	a, b, d	a, b, d

Fuente: esta investigación.

A partir del Anexo A, se realizó la búsqueda de proyectos que desarrollaron software, encontrando 72 (Ver Anexo B) de los cuales, el 44,4% (equivalente a 32 proyectos) pertenecen a la modalidad de aplicativo, 36,1% (equivalente a 26 proyectos) son de pasantía, 19,4% (equivalente a 14 proyectos) corresponden a investigación y ninguno de auditoría. En la Fig. 7, se observa la cantidad de proyectos que superaron cada uno de los filtros y aquellos descartados.

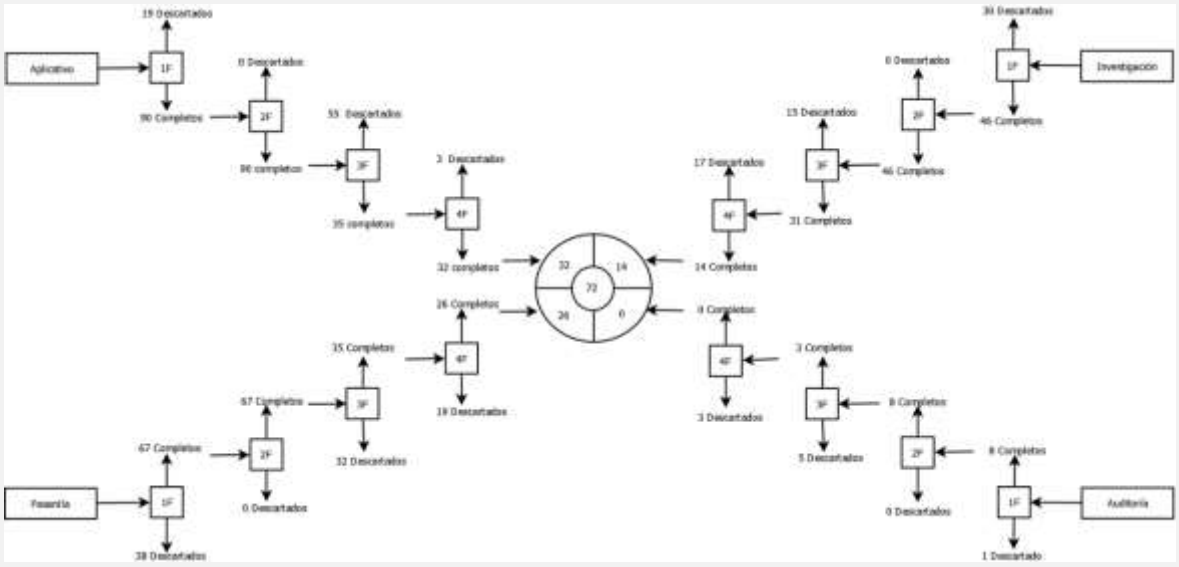


Fig. 7. Mapeo sistemático de aplicación de filtros a proyectos de grado.

Fuente: esta investigación.

3) Evaluar la calidad de los informes de trabajo de grado

Posterior al proceso de selección, se realiza una nueva tarea para asegurar la calidad de los documentos de trabajos de grado encontrados, que corresponde a inspeccionar un conjunto de criterios que se muestran en la TABLA IV.

TABLA IV. CRITERIOS DE EVALUACIÓN DE CALIDAD

Criterio	Descripción	Categoría
C1	Existe documentación completa sobre el análisis realizado al producto software. (Criterios de aceptación o caminos de excepción, como elementos a inspeccionar en la fase de pruebas)	Calidad del reporte
C2	Existe documentación completa sobre la arquitectura del producto software. (En la representación de la arquitectura deberían estar las pruebas, se debe buscar en el diseño)	Calidad del reporte
C3	Existe documentación completa sobre las pruebas realizadas al producto software.	Calidad del reporte
C4	Existe documentación sobre los escenarios de prueba.	Relevancia
C5	Existen resultados de la ejecución de las pruebas.	Credibilidad

Fuente: esta investigación.

Para evaluar la calidad de los documentos se estableció una escala para inspeccionar el nivel de cumplimiento de los criterios, de la siguiente manera: Alto (2 puntos), Medio (1 punto) y Bajo (0 puntos). Los documentos que cumplieron con una valoración igual o superior al 60%, del total de los puntos posibles, son lo que finalmente se eligieron. En el Anexo C, se puede observar la valoración de todos los trabajos seleccionados.

TABLA V. PROYECTOS QUE SUPERARON LOS CRITERIOS DE CALIDAD.

Proyecto	C1	C2	C3	C4	C5	Total	%FO
SADAPACS - SISTEMATIZACIÓN DEL ARCHIVO DE DOCUMENTOS DE LOS ASPIRANTES AL PROGRAMA DE VÍCTIMAS DE LA VIOLENCIA EN ACCIÓN SOCIAL - PASTO NARIÑO	1	1	2	1	2	7	70%
SIRSUB SOFTWARE PARA LA ADMINISTRACIÓN Y DEPURACIÓN Y DE BASE DE DATOS DEL REGIMEN SUBDIADO DEL MUNICIPIO DE TUMACO	2	1	2	1	1	7	70%
TECNOSOFT - SOFTWARE DE GESTION TECNOLOGICA HOSPITALARIA PARA EL HOSPITAL CIVIL DE IPIALES.	1	0	2	2	2	7	70%
IMPLEMENTACION DE UNA SOLUCION INFORMATICA PARA FACTURACION Y RECAUDO DEL IMPUESTO PREDIAL, UNIFICADO, COMO MODULO INTEGRADO AL APLICATIVO: COMPUTACONTA	1	1	2	1	1	6	60%

Proyecto	C1	C2	C3	C4	C5	Total	%FO
SIGLHA-SISTEMA DE INFORMACION DE GLOBAL HUMANITARIA PARA EL MANEJO DEL PROGRAMA DE APADRINAMIENTO EN COLOMBIA	0	2	2	1	1	6	60%
KNOWER - AMBIENTE VIRTUAL PARA LA GESTIÓN DE CONOCIMIENTO PARA INTEGRAR LA LABOR ACADÉMICA E INVESTIGATIVA DE LA FACULTAD DE INGENIERIA DE LA UNIVERSIDAD DE NARIÑO FASEO II- OPTIMIZACION Y PUESTA EN FUNCIONAMIENTO.	2	2	1	1	1	7	70%
VALIDACIÓN DE LA APLICABILIDAD DE PEGASO MEDIANTE EL DESARROLLO DE UN SOFTWARE PARA GESTION DE BIBILIOTECA	2	1	2	1	2	8	80%
SISTEMA DE INFORMACION DEL FONDO DE SEGURIDAD SOCIAL EN SALUD DE LA UNIVERSIDAD DE NARIÑO, MODULO DE APOYO A PROGRAMAS DE PROMOCION Y PREVENCION DE LA SALUD	2	1	2	1	1	7	70%
AWAD: DESARROLLO E IMPLEMENTACION DE APLICACIÓN WEB COMO CANAL DE COMUNICACIÓN ENTRE LA PYME ALPINITA Y SUS CLIENTES DOMICILIADOS	2	1	1	1	1	6	60%
SICEV APLICATIVO ORIENTADO A LA WEB PARA GESTIONAR LA INFORMACION DE LAS VICTIMAS EN LA UNIDAD DE ATENCION Y REPARACION INTEGRAL A LAS VICTIMAS-ERRITORIAL NARIÑO	1	1	2	1	1	6	60%
MODULOS DE PACIENTE SEGURO (CONTROL DE EVENTOS ADVERSOS) Y PETICIONES, QUEJAS, RECLAMOS, SUGERENCIAS Y FELICITACIONES (PQRSF) PARA EL SOFTWARE COMPUCONTA	1	2	2	1	1	7	70%
SKOR SOFTWARE PARA LA GESTION DE ASIGNACIN DE PUNTAJE A DOCENTES TIEMPO COMPLETO DE LA UNIVERSIDAD DE NARIÑO	1	2	2	1	1	7	70%
SAIN, DISEÑO, DESARROLLO E IMPLEMENTACION DE UN SISTEMA DE INFORMACION PARA EL CONTROL Y REGITRO DE NOTAS DEL LICEO INTEGRADO DE BACHILLERATO DE LA UNIVERSIDAD DE NARIÑO	1	2	2	1	1	7	70%
FENIX - SOFTWARE DE APOYO PARA LA IMPLEMETACION DE BALANCED SCORECARD EN UNIDADES ACADEMICAS Y ADMINISTRATIVAS DE LA UNIVERSIDAD DE NARIÑO	1	2	1	1	1	6	60%

Proyecto	C1	C2	C3	C4	C5	Total	%FO
SOFTWARE DE ALMACEN PARA EL MANEJO DE PROPIEDAD PLANTA Y EQUIPO E INSUMOS DE LA UNIVERSIDAD DE NARIÑO ACOPLADO A LAS NIIF	1	2	1	1	1	6	60%
ADM - RIS, SOFTWARE DE APOYO PARA LA ADMINISTRACION DE RIESGOS EN LAS UNIDADES ACADEMICAS Y ADMINISTRATIVAS DE LA UNIVERSIDAD DE NARIÑO	1	1	2	1	2	7	70%
DESARROLLO E IMPLEMENTACION DE UNA PLATAFORMA WEB PARA EL SISTEMA DE CONTROL ADMINISTRATIVO DE RECURSOS EN LA INSTITUCION EDUCATIVA FAUSTINO ARIAS REYNEL	2	1	1	1	1	6	60%
FINDROUT, APLICACIÓN MOVIL PARA EL APOYO EN EL PROCESO DE SELECCIÓN DE RUTA DE TRANSPORTE PUBLICO EN EL MUNICIPIO DE IPIALES	1	1	2	1	2	7	70%
BUSINESS CENSUS, SISTEMA DE INFORMACION DE CENSO EMPRESARIAL PARA LA CAMARA DE COMERCIO DE LA CIUDAD DE IPIALES, MODULO DE CONSULTA Y REPORTEZ	1	1	2	1	1	6	60%
LOBOT - UN ROBOT PARA EL DESARROLLO DE HABILIDADES METACOGNITIVAS EN NIÑOS ENTRE LOS 7 Y 9 AÑOS DE EDAD	2	1	1	1	1	6	60%
SITAPP: UNA APLICACIÓN INTELIGENTE PARA DISPOSITIVOS MOVILES DEL SISTEMA DE TRANPORTE URBANO DEL MUNICIPIO DE PASTO	1	1	2	1	1	6	60%
AYLLU, HERRAMIENTA DE APOYO PARA LA GESTION DE LA INFORMACIN DEL ESTADO DE CONSERVACION DEL QHAPAQÑAN SISTEMA VIAL ANDINO PATRIMONIO CULTURAL DE LA HUMANDAD (COLOMBIA)	1	2	1	1	1	6	60%
DESARROLLO DE UNA PLATAFORMA COMPUTACIONAL PARA LA GESTION DOCUMENTAL DEL SISTEMA DE CALIDAD BASADO EN LA NORMA NTC ISO 9001:2015 EN LA EMPRESA GRUPO BYZA S.A.S.	1	1	2	1	2	7	70%

Proyecto	C1	C2	C3	C4	C5	Total	%FO
"OPTIMIZACIÓN E IMPLEMENTACIÓN DE NUEVAS FUNCIONALIDADES DEL SISTEMA DE INFORMACIÓN DE EVALUACIÓN GENÉTICA DEL PROYECTO DE INVESTIGACIÓN “SELECCIÓN MEDIANTE MODELOS GENÓMICOS Y POLIGÉNICOS PARA EL MEJORAMIENTO GENÉTICO DE LOS BOVINOS DE LECHE EN EL TRÓPICO ALTO DE NARIÑO”"	2	1	1	1	2	7	70%
MALPIN - SISTEMA PARA LA MOTIVACION Y AFIANZAMIENTO DE LA LECTURA EN LA PRIMERA INFANCIA	2	1	2	2	2	9	90%
EMEESA - SOFTWARE PARA EL MAHEJO DE INFORMACION DEL AREA DE DISTRIBUCION DE LA EMPRESA MUNICIPAL DE ENERGIA ELECTRICA S.A E.S.P. DE LA CIUDAD DE POPAYAN	1	1	2	1	1	6	60%
CW - TEAMS: SOFTWARW PARA LA CONFORMACION DE GRUPOS DE TRABAJO COLABOORATIVO BASADOS EN ALGORITMOS GENETICOS	2	1	2	1	1	7	70%
ACTUALIZACION DEL SITIO WEB CIOHP PARA LA GESTION Y VISUALIZACION DE LA INFORMACION DE LAS VARIABLES METEREOLOGICAS DEL PACIFICO COLOMBIANO	2	1	1	1	1	6	60%
GRAVSOFT - SOFTWARE ORIENTADO A LA WEB PARA LA GESTION DE RESERVAS DE AUDITORIOS Y VIDEOCONFERENCIAS EN EL CENTRO DE INVESTIGACIONES OCEANOGRAICAS E HIDROGRAFICAS DEL PACIFICO	2	1	1	1	1	6	60%
SISTEMA DE INFORMACION ORIENTADO A PLATAFORMAS WEB Y WEB MOVIL PARA LA CONSULTA, MODIFICACION, ACTUALIZACION Y DIFUSION DE LA NORMATIVIDAD LEGAL DE LA UNIVERSIDAD DE NARIÑO APLICANDO HASTA EL NIVEL DE CONFORMIDAD AA DE LA NORMA TECNICA COLOMBIANA DE ACCESIBILIDAD WEB NTC 5854	1	1	2	2	1	7	70%
SICAR - SISTEMA DE INFORMACION ORIENTADO A LA WEB PARA FORTALECER EL CENTRO DE DOCUMENTACION E LOS ARTISTAS DEL CARNAVAL DE NEGROS Y BLANCOS DE PASTO	1	1	2	1	2	7	70%

Proyecto	C1	C2	C3	C4	C5	Total	%FO
SISMAR - SISTEMA DE INFORMACION WEB PARA LA GESTION DE REGISTRO, CONTROL DE HOSPEDAJE, MANEJO DE INVENTARIOS Y FACTURACION EN EL HOTEL MAR AZUL	1	2	2	1	1	7	70%

Fuente: esta investigación.

Finalmente, en la TABLA VI, se pueden observar los proyectos que serán analizados en esta investigación.

TABLA VI. PROYECTOS SELECCIONADOS PARA EL ANÁLISIS.

Id	Proyecto
D01	SADAPACS - SISTEMATIZACIÓN DEL ARCHIVO DE DOCUMENTOS DE LOS ASPIRANTES AL PROGRAMA DE VÍCTIMAS DE LA VIOLENCIA EN ACCIÓN SOCIAL - PASTO NARIÑO
D02	SIRSUB SOFTWARE PARA LA ADMINISTRACIÓN Y DEPURACIÓN Y DE BASE DE DATOS DEL REGIMEN SUBDIADO DEL MUNICIPIO DE TUMACO
D03	TECNOSOFT - SOFTWARE DE GESTION TECNOLOGICA HOSPITALARIA PARA EL HOSPITAL CIVIL DE IPIALES.
D04	IMPLEMENTACION DE UNA SOLUCION INFORMATICA PARA FACTURACION Y RECAUDO DEL IMPUESTO PREDIAL, UNIFICADO, COMO MODULO INTEGRADO AL APLICATIVO: COMPUTACONTA
D05	SIGLHA-SISTEMA DE INFORMACION DE GLOBAL HUMANITARIA PARA EL MANEJO DEL PROGRAMA DE APADRINAMIENTO EN COLOMBIA
D06	KNOWER - AMBIENTE VIRTUAL PARA LA GESTIÓN DE CONOCIMIENTO PARA INTEGRAR LA LABOR ACADÉMICA E INVESTIGATIVA DE LA FACULTAD DE INGENIERIA DE LA UNIVERSIDAD DE NARIÑO FASEO II- OPTIMIZACION Y PUESTA EN FUNCIONAMIENTO.
D07	VALIDACIÓN DE LA APLICABILIDAD DE PEGASO MEDIANTE EL DESARROLLO DE UN SOFTWARE PARA GESTION DE BIBILIOTECA

Id	Proyecto
D08	SISTEMA DE INFORMACION DEL FONDO DE SEGURIDAD SOCIAL EN SALUD DE LA UNIVERSIDAD DE NARIÑO, MODULO DE APOYO A PROGRAMAS DE PROMOCION Y PREVENCION DE LA SALUD
D09	AWAD: DESARROLLO E IMPLEMENTACION DE APLICACIÓN WEB COMO CANAL DE COMUNICACIÓN ENTRE LA PYME ALPINITA Y SUS CLIENTES DOMICILIADOS
D10	SICEV APLICATIVO ORIENTADO A LA WEB PARA GESTIONAR LA INFORMACION DE LAS VICTIMAS EN LA UNIDAD DE ATENCION Y REPARACION INTEGRAL A LAS VICTIMAS-ERRITORIAL NARIÑO
D11	MODULOS DE PACIENTE SEGURO (CONTROL DE EVENTOS ADVERSOS) Y PETICIONES, QUEJAS, RECLAMOS, SUGERENCIAS Y FELICITACIONES (PQRSF) PARA EL SOFTWARE COMPUCONTA
D12	SKOR SOFTWARE PARA LA GESTION DE ASIGNACIN DE PUNTAJE A DOCENTES TIEMPO COMPLETO DE LA UNIVERSIDAD DE NARIÑO
D13	SAIN, DISEÑO, DESARROLLO E IMPLEMENTACION DE UN SISTEMA DE INFORMACION PARA EL CONTROL Y REGITRO DE NOTAS DEL LICEO INTEGRADO DE BACHILLERATO DE LA UNIVERSIDAD DE NARIÑO
D14	FENIX - SOFTWARE DE APOYO PARA LA IMPLEMETACION DE BALANCED SCORECARD EN UNIDADES ACADEMICAS Y ADMINISTRATIVAS DE LA UNIVERSIDAD DE NARIÑO
D15	SOFTWARE DE ALMACEN PARA EL MANEJO DE PROPIEDAD PLANTA Y EQUIPO E INSUMOS DE LA UNIVERSIDAD DE NARIÑO ACOPLADO A LAS NIIF
D16	ADM - RIS, SOFTWARE DE APOYO PARA LA ADMINISTRACION DE RIESGOS EN LAS UNIDADES ACADEMICAS Y ADMINISTRATIVAS DE LA UNIVERSIDAD DE NARIÑO
D17	DESARROLLO E IMPLEMENTACION DE UNA PLATAFORMA WEB PARA EL SISTEMA DE CONTROL ADMINISTRATIVO DE RECURSOS EN LA INSTITUCION EDUCATIVA FAUSTINO ARIAS REYNEL
D18	FINDROUT, APLICACIÓN MOVIL PARA EL APOYO EN EL PROCESO DE SELECCIÓN DE RUTA DE TRANSPORTE PUBLICO EN EL MUNICIPIO DE IPIALES
D19	BUSINESS CENSUS, SISTEMA DE INFORMACION DE CENSO EMPRESARIAL PARA LA CAMARA DE COMERCIO DE LA CIUDAD DE IPIALES, MODULO DE CONSULTA Y REPORTEZ
D20	LOBOT - UN ROBOT PARA EL DESARROLLO DE HABILIDADES METACOGNITIVAS EN NIÑOS ENTRE LOS 7 Y 9 AÑOS DE EDAD
D21	SITAPP: UNA APLICACIÓN INTELIGENTE PARA DISPOSITIVOS MOVILES DEL SISTEMA DE TRANPORTE URBANO DEL MUNICIPIO DE PASTO

Id	Proyecto
D22	AYLLU, HERRAMIENTA DE APOYO PARA LA GESTION DE LA INFORMACIN DEL ESTADO DE CONSERVACION DEL QHAPAQÑAN SISTEMA VIAL ANDINO PATRIMONIO CULTURAL DE LA HUMANDAD (COLOMBIA)
D23	DESARROLLO DE UNA PLATAFORMA COMPUTACIONAL PARA LA GESTION DOCUMENTAL DEL SISTEMA DE CALIDAD BASADO EN LA NORMA NTC ISO 9001:2015 EN LA EMPRESA GRUPO BYZA S.A.S.
D24	"OPTIMIZACIÓN E IMPLEMENTACIÓN DE NUEVAS FUNCIONALIDADES DEL SISTEMA DE INFORMACIÓN DE EVALUACIÓN GENÉTICA DEL PROYECTO DE INVESTIGACIÓN “SELECCIÓN MEDIANTE MODELOS GENÓMICOS Y POLIGÉNICOS PARA EL MEJORAMIENTO GENÉTICO DE LOS BOVINOS DE LECHE EN EL TRÓPICO ALTO DE NARIÑO”"
D25	MALPIN - SISTEMA PARA LA MOTIVACION Y AFIANZAMIENTO DE LA LECTURA EN LA PRIMERA INFANCIA
D26	EMEESA - SOFTWARE PARA EL MAHEJO DE INFORMACION DEL AREA DE DISTRIBUCION DE LA EMPRESA MUNICIPAL DE ENERGIA ELECTRICA S.A E.S.P. DE LA CIUDAD DE POPAYAN
D27	CW - TEAMS: SOFTWARW PARA LA CONFORMACION DE GRUPOS DE TRABAJO COLABOORATIVO BASADOS EN ALGORITMOS GENETICOS
D28	ACTUALIZACION DEL SITIO WEB CIOHP PARA LA GESTION Y VISUALIZACION DE LA INFORMACION DE LAS VARIABLES METEREOLOGICAS DEL PACIFICO COLOMBIANO
D29	GRAVSOFT - SOFTWARE ORIENTADO A LA WEB PARA LA GESTION DE RESERVAS DE AUDITORIOS Y VIDEOCONFERENCIAS EN EL CENTRO DE INVESTIGACIONES OCEANOGRAICAS E HIDROGRAFICAS DEL PACIFICO
D30	SISTEMA DE INFORMACION ORIENTADO A PLATAFORMAS WEB Y WEB MOVIL PARA LA CONSULTA, MODIFICACION, ACTUALIZACION Y DIFUSION DE LA NORMATIVIDAD LEGAL DE LA UNIVERSIDAD DE NARIÑO APLICANDO HASTA EL NIVEL DE CONFORMIDAD AA DE LA NORMA TECNICA COLOMBIANA DE ACCESIBILIDAD WEB NTC 5854
D31	SICAR - SISTEMA DE INFORMACION ORIENTADO A LA WEB PARA FORTALECER EL CENTRO DE DOCUMENTACION E LOS ARTISTAS DEL CARNAVAL DE NEGROS Y BLANCOS DE PASTO
D32	SISMAR - SISTEMA DE INFORMACION WEB PARA LA GESTION DE REGISTRO, CONTROL DE HOSPEDAJE, MANEJO DE INVENTARIOS Y FACTURACION EN EL HOTEL MAR AZUL

Fuente: esta investigación.

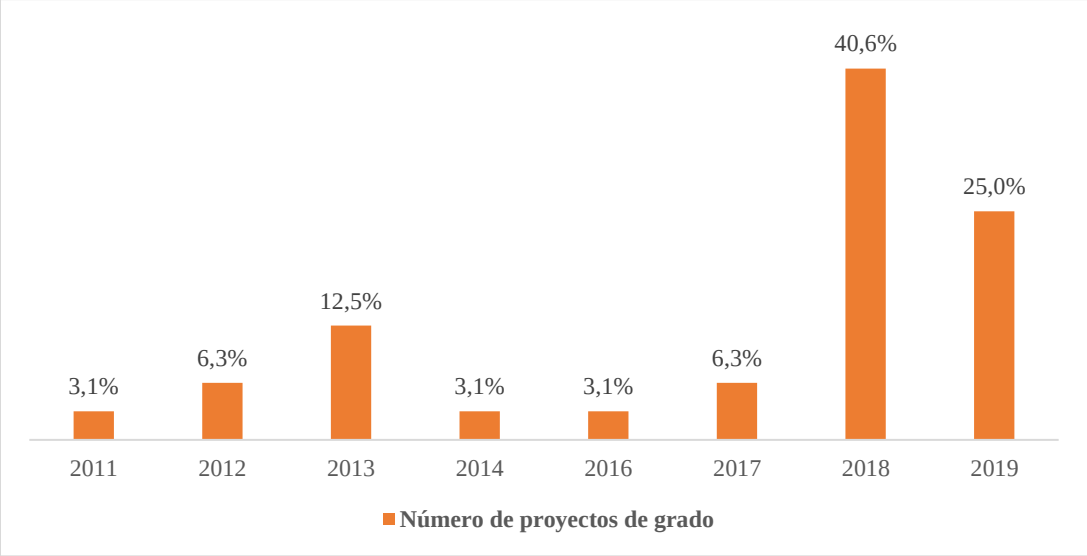
4) Extraer la información de los proyectos

Una vez efectuada la evaluación de calidad de los documentos de grado, se selecciona un total de 32 proyectos, con los cuales se obtiene el Anexo D mediante el registro de información extraída de los mismos.

A partir de los proyectos seleccionados, se observa que el mayor número corresponde al año 2018 con el 40,6% (equivalente a 13 proyectos), seguido del año 2019 con el 25,0% (equivalente a ocho proyectos).

Entre los años 2011 al 2013 se presenta un incremento que parte del 3,1% (equivalente a un proyecto) hasta el 12,5% (equivalente a cuatro proyectos), posteriormente se presenta otro aumento que parte del año 2014 con un 3,1% (equivalente a un proyecto) y termina en el año 2018 con un 40,6% (equivalente a 13 proyectos).

Los documentos de los trabajos desarrollados en los años 2010, 2015 y 2020 son descartados, debido a que no superaron los filtros y criterios descritos en las secciones anteriores.

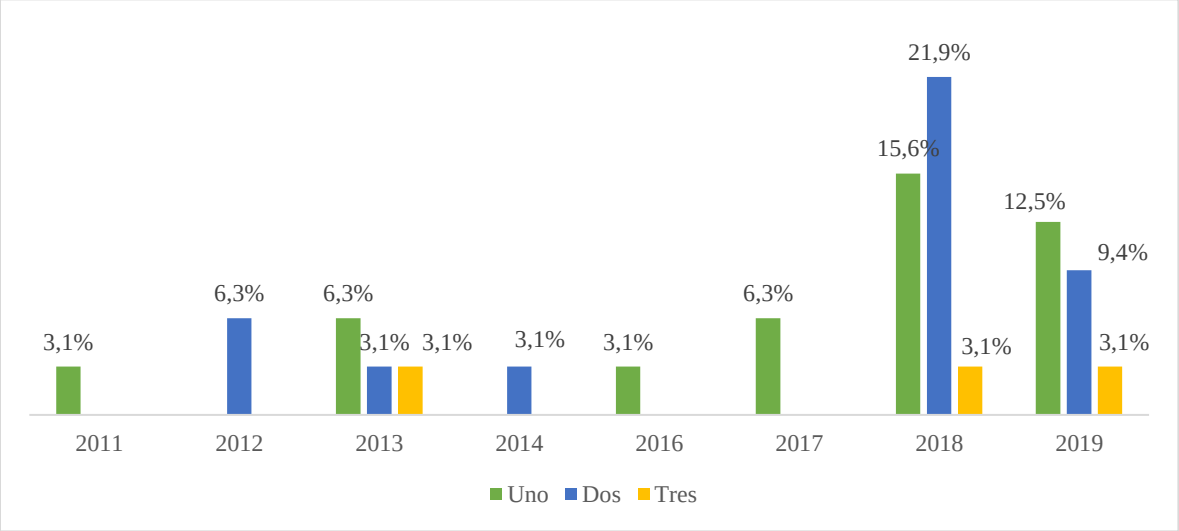


Gráfica 9. Número de proyectos seleccionados por año.

Fuente: esta investigación.

El número de integrantes que conforman los proyectos de grado seleccionados suman 52 autores.

En la Gráfica 10 se visualiza que las categorías de uno y dos autores son las más frecuentes, presentando su punto más alto en el año 2018 con un 15,6% (equivalente a cinco proyectos) y 21,9% (equivalente a siete proyectos) respectivamente; además, se muestra que la categoría de tres autores es la menos usual, con un 3,1% (equivalente a un proyecto) en los años en los que se presenta esta categoría.

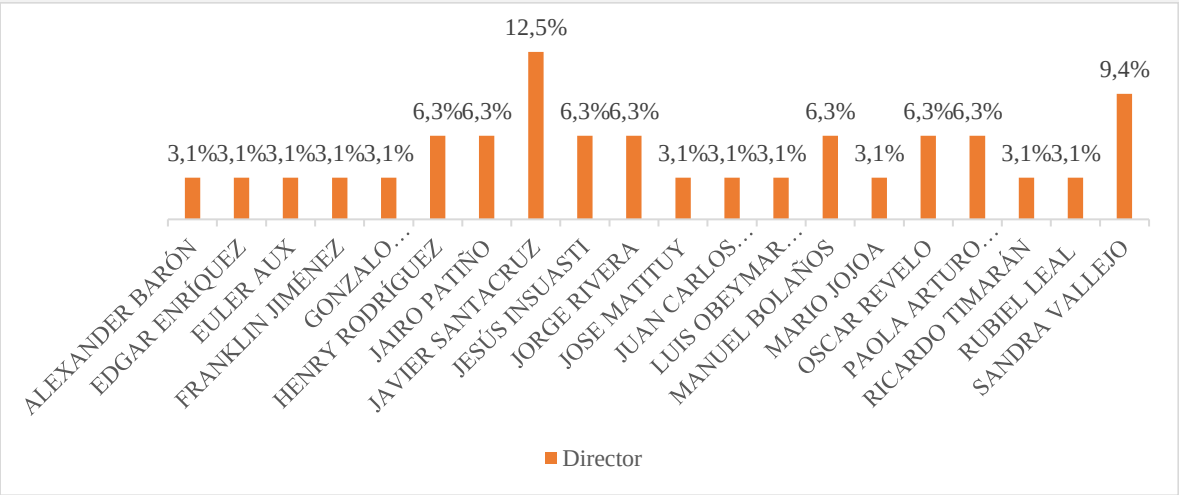


Gráfica 10. Número de autores por proyecto de grado seleccionado según el año.

Fuente: esta investigación.

Los trabajos de grado seleccionados fueron asesorados por 20 docentes como directores y siete docentes como co-directores.

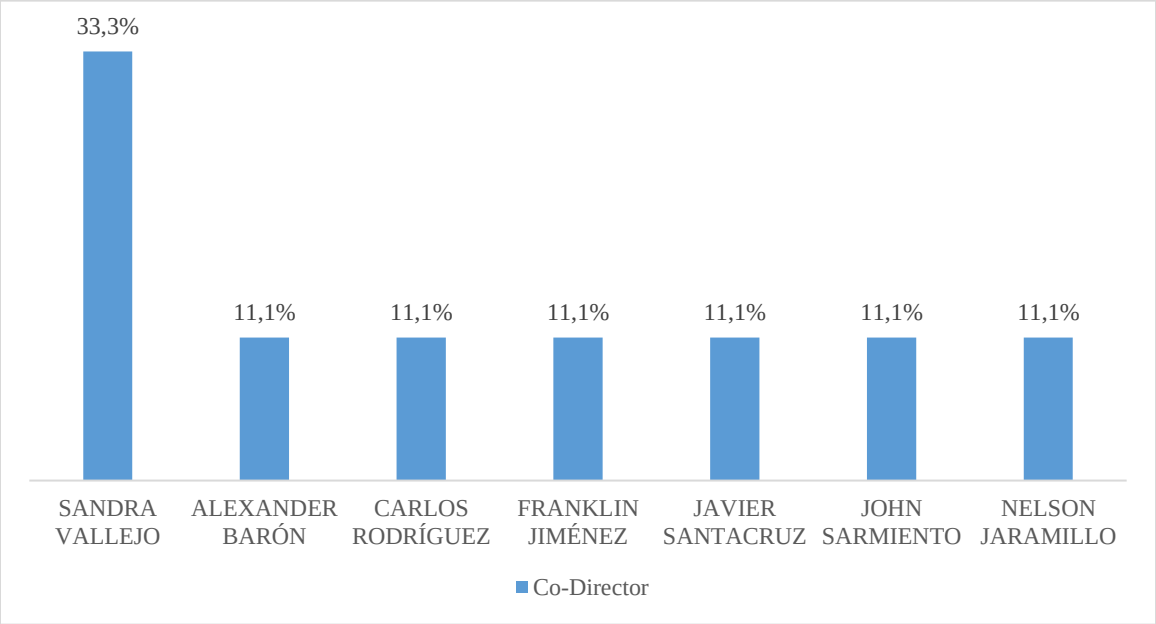
En la Gráfica 11, se observa que el docente con la mayor cantidad de trabajos de grado dirigidos es el ingeniero Javier Santacruz, con una frecuencia del 12,5% (equivalente a cuatro proyectos), mientras que la menor frecuencia es del 3,1% (equivalente a un proyecto), siendo esta la que más se repite.



Gráfica 11. Proyectos de grado seleccionados por director.

Fuente: esta investigación.

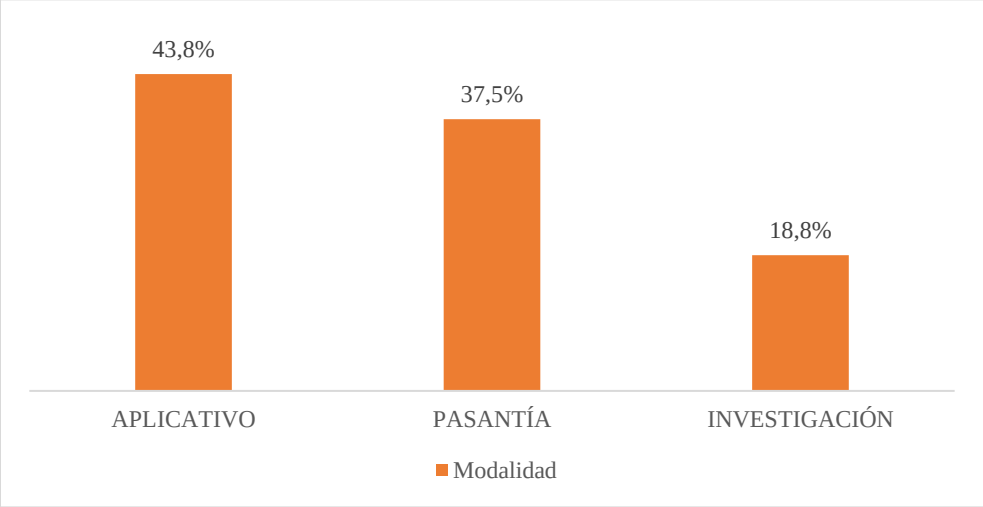
Por otra parte, en aquellos trabajos de grado que tienen un co-director, la ingeniera Sandra Vallejo es quien apoya el mayor número de proyectos de grado, con un 33,3% (equivalente a cinco proyectos); en comparación a la menor frecuencia correspondiente al 11,1% (equivalente a un proyecto), la cual es alcanzada por seis docentes, como se muestra en la Gráfica 12.



Gráfica 12. Co-directores según proyectos de grado aprobados.

Fuente: esta investigación.

Categorizando los trabajos de grado según su modalidad (Ver Gráfica 13), se tiene que el mayor número de proyectos pertenecen a la modalidad de aplicativo, con 14 proyectos que corresponden al 43,8%; seguido de la modalidad de pasantía con 12 proyectos equivalente a 37,5%, y finalmente se tiene la modalidad de investigación con seis proyectos correspondiente al 18,8%. En este caso, ningún proyecto de grado pertenece a la modalidad de auditoría, por lo cual no se presenta en la Gráfica 13.



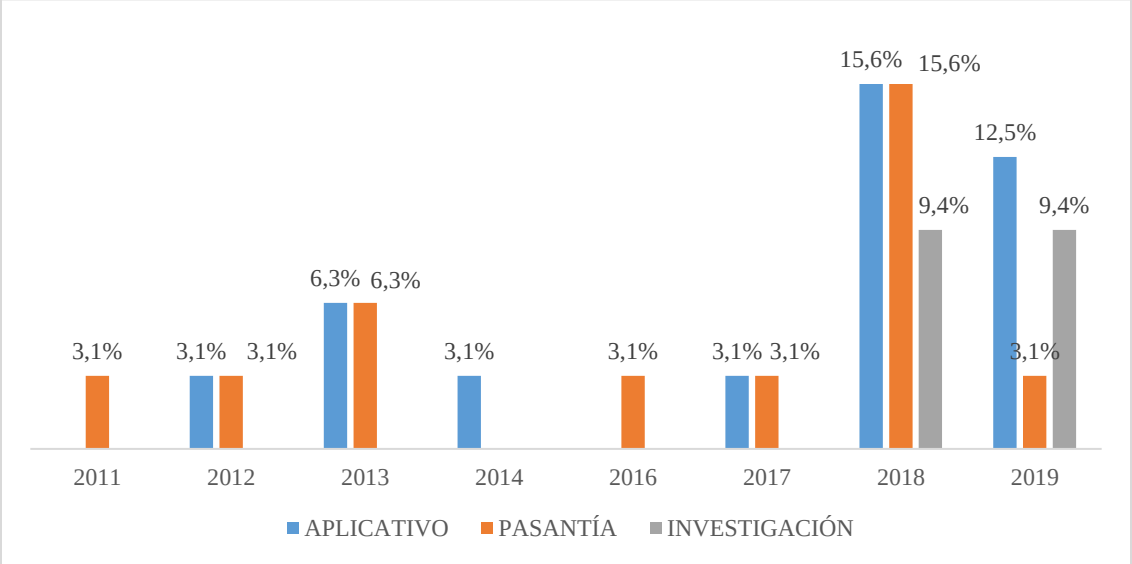
Gráfica 13. Modalidad de los proyectos seleccionados.

Fuente: esta investigación.

La Gráfica 14 muestra los proyectos de grado clasificados por modalidad y distribuidos según su año de finalización.

En el año 2018 las modalidades de aplicativo, pasantía e investigación, alcanzan el porcentaje más alto, respecto al periodo de tiempo trabajado en esta investigación, con el 15,6% (equivalente a cinco proyectos) para la modalidad de aplicativo y pasantía, y el 9,4% (equivalente a tres proyectos) para la modalidad de investigación.

Los proyectos de investigación, se encuentran únicamente en los años 2018 y 2019, en comparación con las modalidades de aplicativo y pasantía que son más frecuentes a lo largo del periodo analizado.



Gráfica 14. Modalidad de los proyectos seleccionados según el año.

Fuente: esta investigación.

De los 32 proyectos seleccionados, se buscó información referente a lo utilizado en la construcción de software en cada uno de ellos. La TABLA VII, muestra los datos recolectados, con su respectiva descripción y los valores que pueden tomar, de acuerdo con lo encontrado en los documentos de los proyectos mencionados anteriormente.

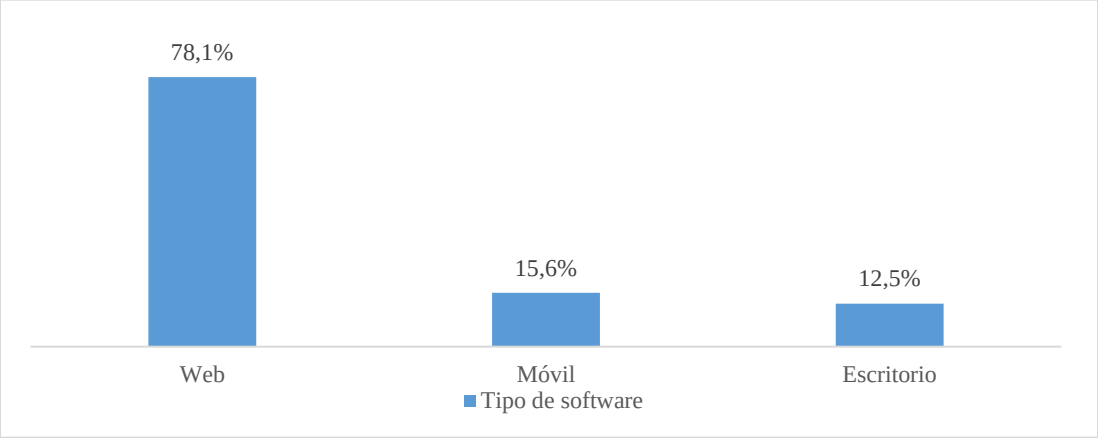
TABLA VII. DATOS DE PROYECTOS APROBADOS.

Información	Descripción	Valores
Tipo	Tipo de software desarrollado en el proyecto de grado.	- Escritorio - Web - Móvil

Información	Descripción	Valores
Tecnología para desarrollo	Herramientas de software utilizadas para el desarrollo del producto software.	**Lenguajes**: lenguajes de programación, cascada o etiqueta, utilizados en la construcción de software, están clasificados en: Escritorio, *Frontend*, *Backend*. Y pueden tomar valores como: Java, C#, Java Script, CSS, HTML, XML, PHP, Ajax, PLSQL. ***Framework***: conjunto de bibliotecas de apoyo para el desarrollo de software, las categorías definidas son: Boostrap, Zend, NodeJS, Express, AngularJS, Laravel, VeuJS, NuxtJS, .NET, Sequelize. **Otro:** herramienta de software empleada en la codificación del software, las categorías fueron: Scriptcase, Netbeans, Android Studio, Eclipse, Visual Studio, SQL Developer, Atom, Visual Studio Code, Sublime Text. **Plataforma:** sistema operativo en que se desarrolló el software, las categorías son: Windows y Linux. **DBMS:** sistemas gestores de bases de datos, las categorías son: PostgreSQL, Microsoft Access, MySQL, Google App Engine, SQLite, SQL Server, Oracle.
Forma	Forma de aplicación de pruebas de software.	- Manual - Automática
Tipo de prueba	Tipo o tipos de pruebas aplicados en el proyecto de grado.	- Aceptación - Unitarias - Integridad - Funcionales - Estrés - Usabilidad - Integración - Interfaz
Tecnologías para la prueba	Herramientas software utilizadas para la aplicación de pruebas automáticas.	- JUnit - Selenium - JMeter

Fuente: esta investigación.

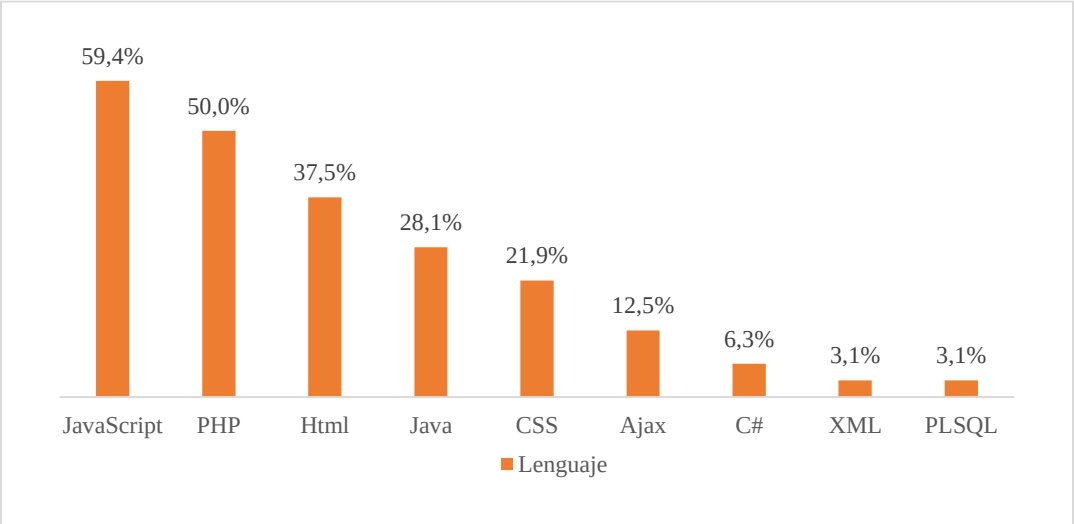
Se distinguen tres tipos de software, los cuales son: escritorio, móvil y web, en la Gráfica 15, se observa que más de la mitad de los proyectos de software son de tipo web, alcanzando el 78,1% (equivalente a 25 proyectos).



Gráfica 15. Tipos de software.

Fuente: esta investigación.

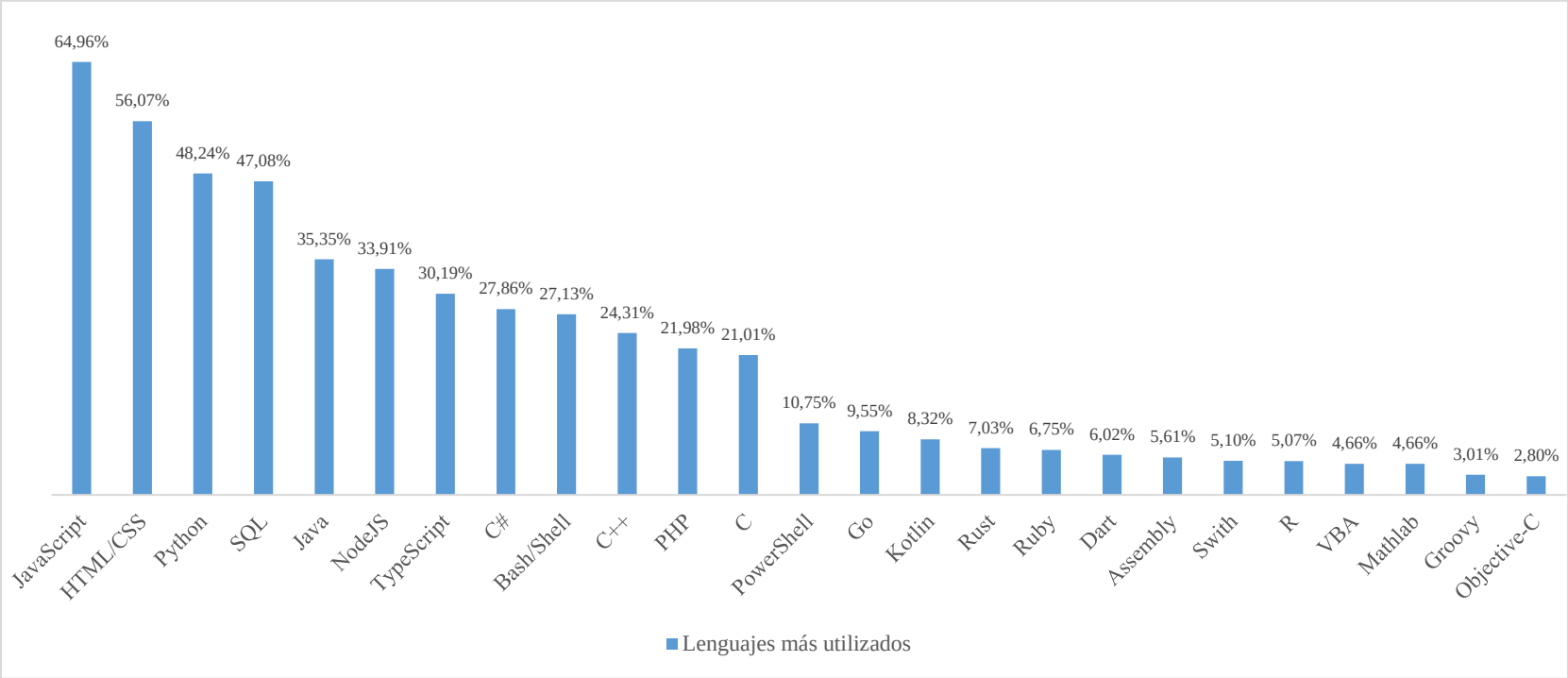
Al realizar una inspección del uso de los lenguajes de: programación, cascada, etiqueta y marcado; en los diferentes proyectos de grado, se encontró que el 96,9% (correspondiente a 31 proyectos) presentan documentación referente a esta categoría, encontrando que algunos proyectos utilizaron más de un lenguaje. En la Gráfica 16 se puede visualizar esta información, donde cada porcentaje corresponde al uso de los lenguajes en base en los 32 proyectos seleccionados; mostrando a JavaScript como el más común, con una frecuencia del 59,4% (equivalente a 19 proyectos) y con la menor frecuencia de 3,1% (equivalente a un proyecto) para los lenguajes XML y PLSQL.



Gráfica 16. Lenguajes utilizados para el desarrollo del software.

Fuente: esta investigación.

Al observar el uso de los lenguajes de programación, etiqueta, cascada o marcado, empleados por desarrolladores a nivel mundial en el año 2021 (Ver Gráfica 17), se observa que, de 83.052 desarrolladores de software, el 64,96% (aproximadamente 53.951 encuestados) tienen preferencia por JavaScript, tendencia que también se ve reflejada en la Universidad de Nariño (Ver Gráfica 16), además, se evidencia que el interés por el desarrollo web, ha estado presente desde el año 2010; manteniéndose hasta la actualidad.

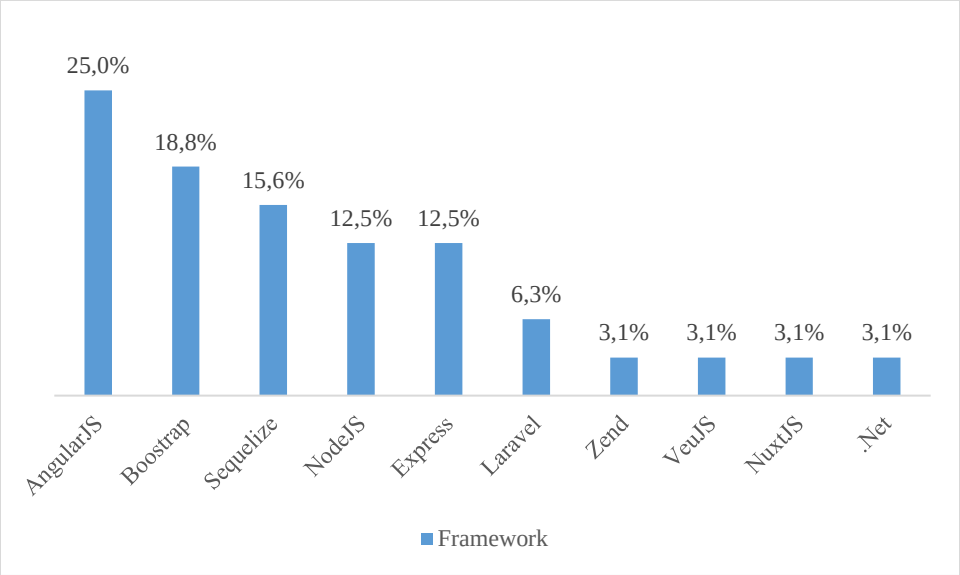


Gráfica 17. Lenguajes de programación más utilizados a nivel mundial, en el año 2021.

Fuente: adaptación hecha de [25]

Analizando el uso de *Frameworks*, se logra identificar que únicamente son utilizados en los proyectos de desarrollo web que conforman el 78,1% (equivalente a 25 proyectos) de los 32 seleccionados (Ver Grafica 15), de los cuales el 40,6% (equivalente a 13 proyectos), hacen uso de uno o más *Frameworks.*

En la Gráfica 18, se puede observar que los *Frameworks* orientados a *Frontend* lideran, siendo AngularJS con un 25,0% (equivalente a ocho proyectos) el más utilizado.

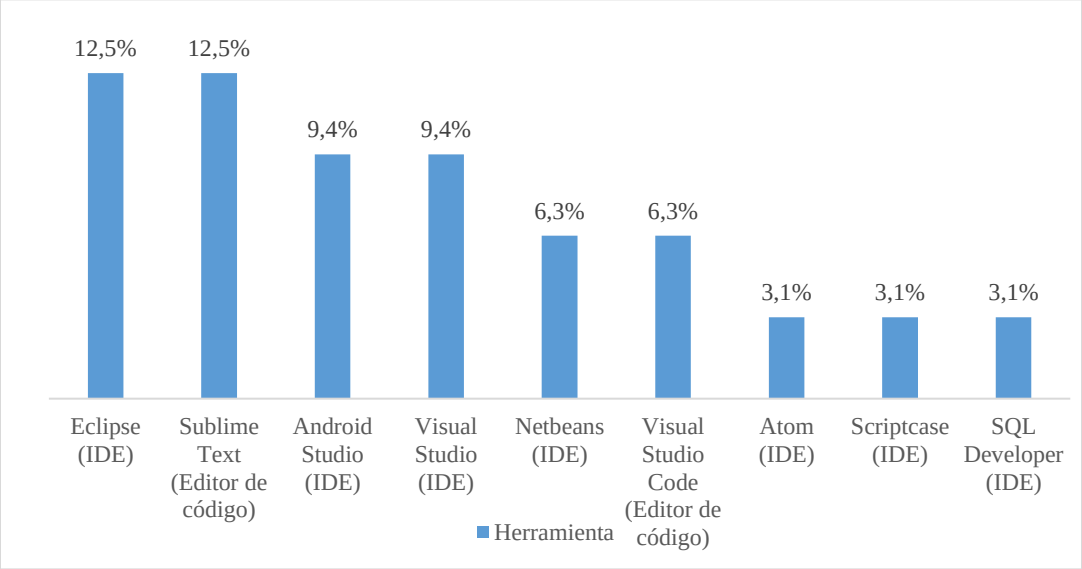


Gráfica 18. *Frameworks* utilizados en los proyectos de grado seleccionados.

Fuente: esta investigación.

Para las herramientas de codificación el 40,6% (equivalente a 13 proyectos) de los proyectos de grado seleccionados, presentan información sobre el uso de uno o más entornos de desarrollo o editores de código.

La Gráfica 19 muestra el uso que tuvieron las herramientas de codificación; siendo el ambiente de desarrollo integrado *(*IDE por sus siglas en inglés*)* Eclipse y el editor de código Sublime Text, las herramientas más empleadas, con una frecuencia del 12,5% (equivalente a cuatro proyectos).

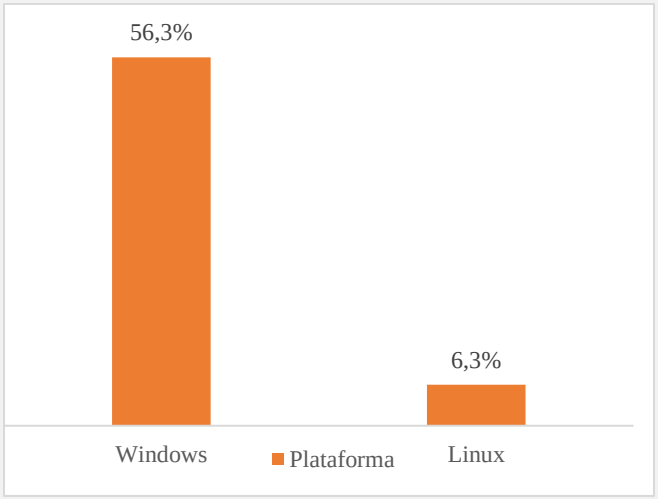


Gráfica 19. Herramientas de codificación.

Fuente: esta investigación.

Los proyectos que especifican la plataforma en que se desarrolló el software corresponden al 62,5% (equivalente a 20 proyectos). Se presentan dos plataformas: Windows y Linux.

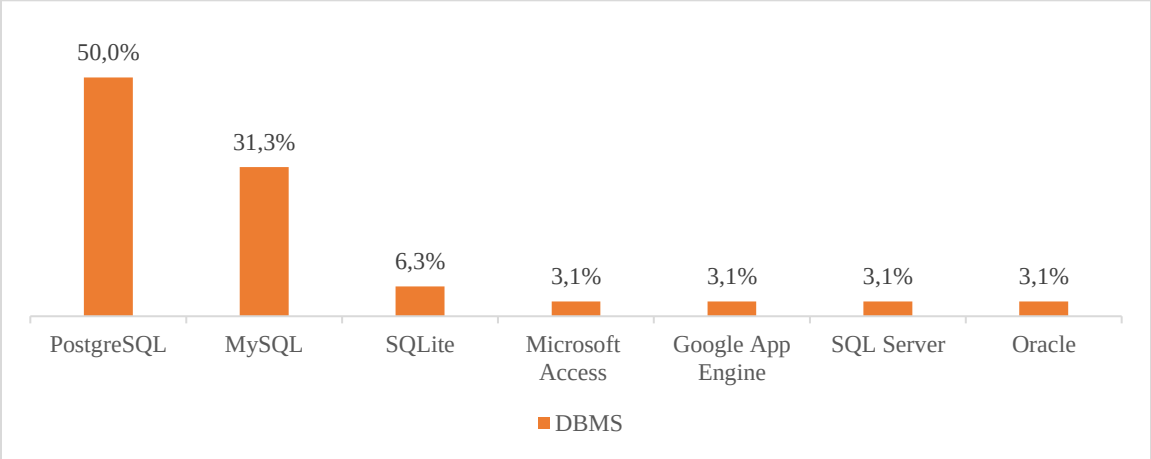
En la Gráfica 20, se puede observar que Windows lidera con el 56,3% (equivalente a 18 proyectos).



Gráfica 20.Plataforma en que se desarrolla el software.

Fuente: esta investigación.

Todos los proyectos de grado hicieron uso de un sistema gestor de base de datos (DBMS por sus siglas en inglés), en la Gráfica 21, se puede observar que PostgreSQL alcanza la frecuencia mayor, con un 50,0% (equivalente a 16 proyectos).

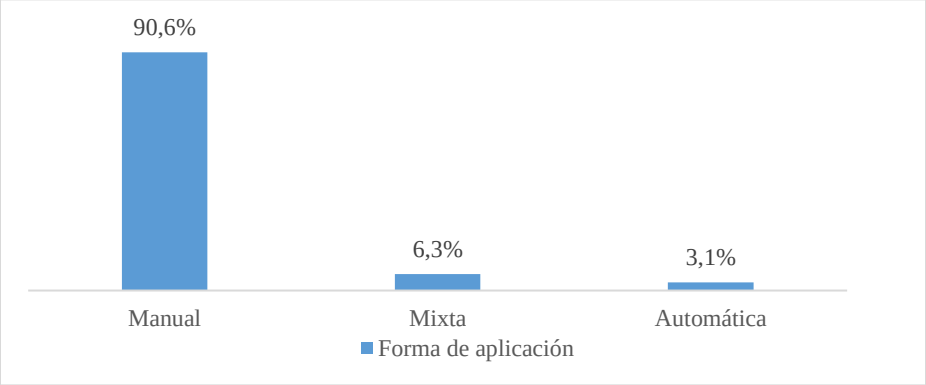


Gráfica 21. Gestores de bases de datos (DBMS)

Fuente: esta investigación.

Para la realización de pruebas de software, se encontró tres formas de aplicación de pruebas: manual, mixta y automática, donde la forma mixta integra pruebas manuales y automáticas. En la Gráfica 22 se observa que el 90,6% (equivalente a 29 proyectos) efectuaron pruebas manuales, y solo un proyecto efectuó pruebas totalmente automáticas que corresponde al 3,1%.

Las herramientas que se identificaron para la automatización de pruebas son: JUnit para pruebas unitarias con el 3,1% (equivalente a un proyecto), Selenium para pruebas de integración e interfaz con una frecuencia del 3,1% (equivalente a un proyecto) y JMeter para pruebas de estrés, siendo esta última herramienta la más utilizada con el 6,2% (equivalente a dos proyectos).

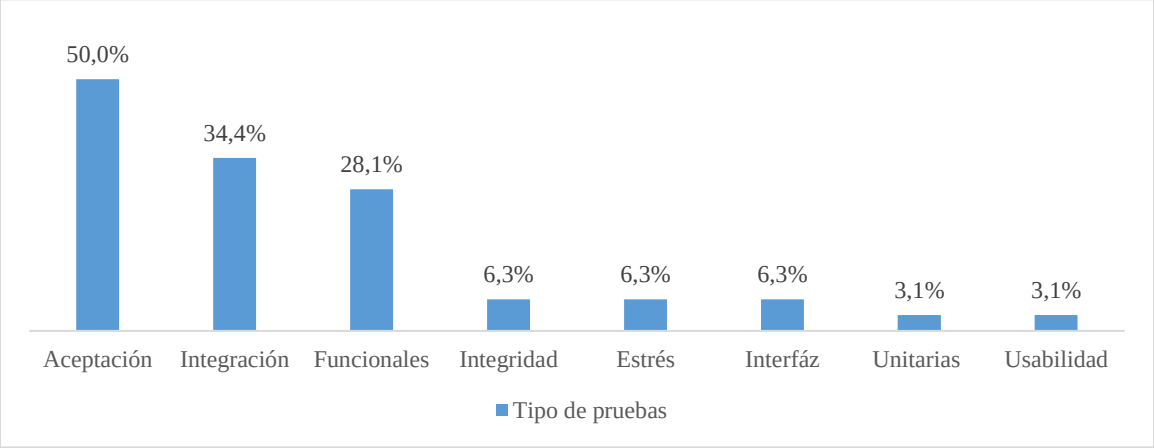


Gráfica 22. Forma de aplicación de pruebas.

Fuente: esta investigación.

Al inspeccionar la fase de pruebas del software que desarrollaron los proyectos de grado, se evidencian ocho tipos de pruebas: Aceptación, Unitarias, Integridad, Funcionales, Estrés, Usabilidad, Integración e Interfaz; de los cuales, las pruebas de tipo aceptación son las más utilizadas, con un 50,0% (equivalente a 16 proyectos de grado), esta información se puede visualizar en la Gráfica 23.

Una anotación a considerar es que, en el 28,1% (equivalente a nueve proyectos) se menciona el uso de pruebas unitarias; sin embargo, al realizar una inspección a la sección del informe final dónde se describe la forma como se desarrollan las pruebas, se evidencia que, en realidad se evalúa el funcionamiento de cada módulo y su integración entre componentes, lo que correspondería al desarrollo de pruebas de integración, ya que, las pruebas unitarias se efectúan de forma automática mediante el uso de una herramienta, para comprobar el funcionamiento de los métodos de una clase; por esta razón, las pruebas de dichos documentos, fueron incluidas en la categoría de pruebas de integración. (Ver Gráfica 23).



Gráfica 23. Tipo de pruebas.

Fuente: esta investigación.

Después de efectuar el proceso sistemático para analizar e identificar la forma como se realizan las pruebas de unidad y componentes, se puede establecer que de 32 proyectos seleccionados en esta investigación; y que están relacionados con la construcción de software en un periodo de realización comprendido entre los años 2010 al 2020, únicamente el 3,1% (equivalente a un proyecto) realizó pruebas unitarias y no se logró evidenciar el desarrollo de pruebas de componentes.

5) Sintetizar resultados

En este capítulo, se caracterizó la forma como se realizan las pruebas de unidad y componentes, en los proyectos de grado de la Universidad de Nariño entre los años 2010 a 2020. Se identificaron dos fuentes de información: el repositorio virtual de la biblioteca de la Universidad de Nariño (SIRED) y el repositorio de la secretaría del Departamento de Sistemas; de las cuales se obtuvo una población de 289 proyectos de grado.

Mediante la aplicación de criterios de inclusión y exclusión, se asegura que el proyecto de grado tenga una fecha de finalización y se encuentre en el rango del periodo trabajado, encontrando que el 27,0% (equivalente a 78 proyectos) no cuentan con una fecha de finalización, por lo cual, se infiere que, al mes de mayo de 2021, es probable que dichos proyectos no hayan culminado. Así mismo, los criterios garantizan que el proyecto de grado cuente con un informe final y haya desarrollado un software.

Posteriormente, se aplican cuatro filtros que, basados en los criterios de inclusión y exclusión, redujeron la población a 72 proyectos que desarrollan software.

Al evaluar la calidad del documento de los proyectos de grado, a través de cinco criterios, se busca que contengan un informe completo respecto al análisis, documentación de arquitectura y pruebas (tanto en escenarios como resultados) del software, obteniendo 32 proyectos que superaron los criterios de evaluación de calidad.

De los 32 proyectos identificados, se observa que la mayor cantidad pertenecen al año 2018 con el 40,6% (equivalente a 13 proyectos), mientras que los proyectos de los años 2010, 2015 y 2020, no superaron los filtros; además, se buscó información relacionada con lo utilizado en la construcción del software, clasificando esta información en las categorías: Tipo, Tecnología de desarrollo, Forma, Tipo de prueba y Tecnologías para la prueba.

A partir de la relación obtenida, se tiene que en el periodo analizado, los proyectos de grado que desarrollaron software, se inclinaron por: el tipo de software orientado a la Web con un 78,1% (equivalente a 25 proyectos), el lenguaje de programación JavaScript con el 59,4% (equivalente a 19 proyectos), *Frameworks* de desarrollo Web siendo el más popular AngularJS con un 25,0% (equivalente a ocho proyectos) y las herramientas de codificación Eclipse y Sublime Text con un 12,5% (equivalente a cuatro proyectos) respectivamente.

En la inspección de la fase de pruebas, se obtuvo que el 90,6% (equivalente a 29 proyectos) realizan las pruebas manualmente y solo el 3,1% (equivalente a un proyecto) aplica pruebas totalmente automatizadas, además, de ser el único que efectúa pruebas unitarias. En ningún proyecto se encontró la realización de pruebas de componentes.

Al finalizar este capítulo se evidencia que, en los proyectos de grado de la Universidad de Nariño que desarrollaron software en el periodo de años seleccionado para la investigación, de los 32 proyectos analizados, sólo el 9,4% (equivalente a tres proyectos) efectuaron pruebas automáticas, de los cuales, el 3,1% (equivalente a un proyecto) realizó pruebas completamente automatizadas, corroborando lo que se definió inicialmente en el planteamiento del problema de esta investigación, en el cual se menciona que por falta de conocimiento en el uso de herramientas para realizar pruebas automáticas, la aplicación de estas es mínima, en comparación con las pruebas manuales.

Por otra parte, se encontró como hallazgo que el 28,1% (equivalente a nueve proyectos), menciona el uso de pruebas unitarias, que en realidad corresponden a pruebas de integración, según la información registrada en la fase de pruebas de los documentos, mostrando que se tenía un concepto erróneo de la definición de pruebas unitarias.

B. PROPUESTA PARA EL DESARROLLO Y EJECUCIÓN DE PRUEBAS AUTOMÁTICAS DE UNIDAD Y COMPONENTES DESDE EL DESARROLLO ÁGIL DE SOFTWARE

El propósito de esta sección es presentar como se diseñó una propuesta para el proceso de desarrollo y ejecución de pruebas automáticas de unidad y componentes desde el enfoque ágil de software, con base en la caracterización. Para este fin, se realizaron las fases que se observan en la Fig. 8.

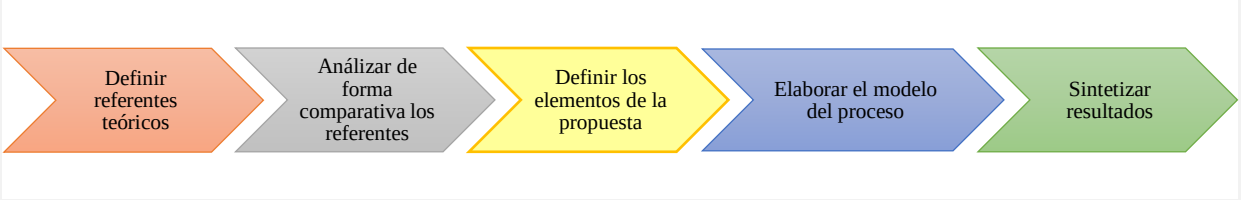


Fig. 8. Proceso del diseño de la propuesta.

Fuente: esta investigación.

1) Definir referentes teóricos para la representación del proceso

Según [26] un proceso es un conjunto de acciones que se llevan a cabo para alcanzar un propósito y se utiliza en muchos ámbitos o contextos.

Dentro del ámbito empresarial u organizacional, para [22] la representación o modelamiento de un proceso, facilita la comprensión del funcionamiento de una entidad o área; ya que permite visualizar los objetivos, metas, actividades y participantes, que interactúan, mostrando la forma en que se desarrollan o ejecutan las acciones, dando la posibilidad de identificar fallas para corregirlas. Además, favorece la comunicación operativa, proporcionando un modelo estándar dónde circula el flujo de información.

Según lo descrito por [22], en el año 2001 se empezó a desarrollar un lenguaje XML para representar los procesos de negocio y su ejecución. Sin embargo, se notó la necesidad de incorporar diagramas para mostrar los procesos gráficamente, de tal manera que para los usuarios sea fácil de entender, ya que sería un lenguaje técnico de representación de procesos estándar. Así mismo, al hacer uso de una herramienta orientada al modelamiento, ésta se adaptaría a la notación especificada.

Actualmente, existe una notación para representar los procesos de negocio de una organización denominada *Business Process Management Notation* - BPMN (BPMN por sus siglas en inglés). Según [22] esta notación permite hacer representaciones de tres tipos: orquestación, coreografía y colaboración. La orquestación es un modelo centralizado, es decir, se basa en representar los procesos de negocio que realiza una entidad. La coreografía es la representación del comportamiento que deben tener los participantes en un proceso de negocio, y la colaboración

representa la comunicación entre los participantes de un proceso de negocio, los cuales pueden estar contenidos en una coreografía.

Haciendo uso de la notación BPMN se diseña la Fig. 9, en la que se presenta un ejemplo del modelado de procesos de negocio. Este modelo corresponde con la forma como se desarrolla la caracterización de los trabajos de grado para analizar la manera como realizan pruebas de unidad y componentes (primer objetivo de esta investigación). Para su elaboración se hizo uso de la herramienta Bizagi Modeler[1].

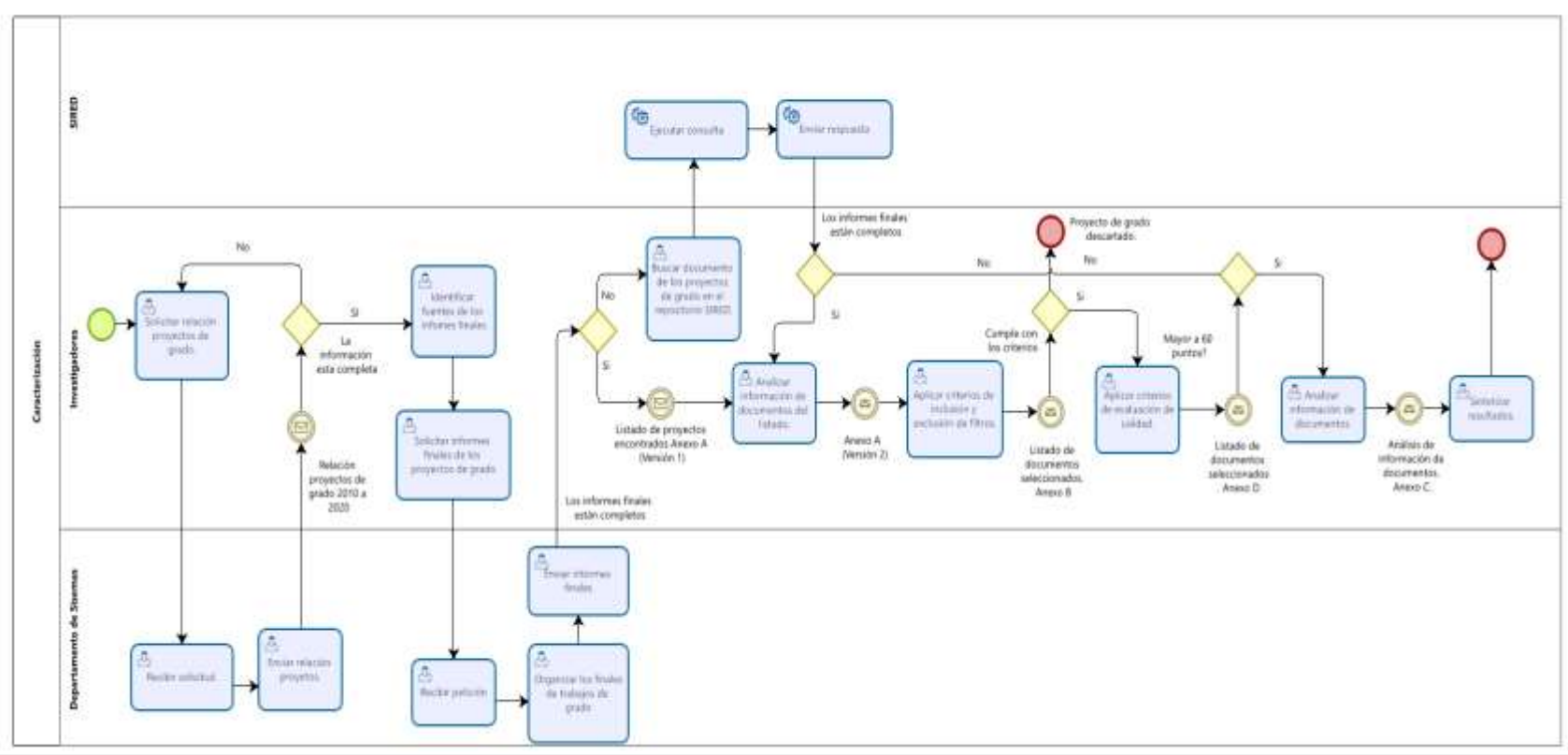


Fig. 9. Modelo del proceso del objetivo uno de la investigación.

Fuente: esta investigación.

Una vez establecida la forma para representar el proceso, se continuó con la búsqueda de formas, caminos o maneras de realizar *Agile Testing*, o pruebas desde el desarrollo ágil de software. Para identificar los métodos más relevantes, se realizó un mapeo sistemático empleando elementos propuestos en el protocolo planteado por [27]. El mapeo consta de las fases que se presentan en la Fig. 10.

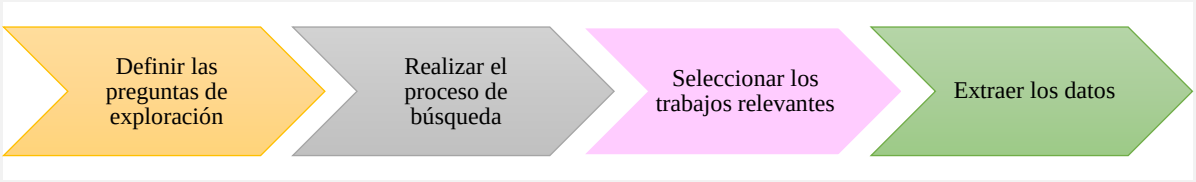


Fig. 10. Proceso para el mapeo sistemático de literatura.

Fuente: esta investigación.

a) Definir preguntas de exploración.

Las preguntas de exploración formuladas se basan en el interrogante principal: **¿Cuáles son los métodos utilizados en la realización de pruebas desde el desarrollo ágil de software?**

A partir de la pregunta principal se derivaron las preguntas secundarias que permitieron analizar y categorizar los referentes:

RQ1. ¿Qué métodos son los más utilizados para realizar pruebas desde el desarrollo ágil de software?

RQ2. ¿Cuáles son los roles que cumplen las personas que intervienen en la realización de pruebas en los métodos encontrados?

RQ3. ¿Cuáles son las fases y actividades de los métodos para la realización de pruebas desde el desarrollo ágil de software?

RQ4. ¿Qué flujos de control se presentan en la ejecución de las fases y actividades de los métodos para la realización de pruebas desde el desarrollo ágil de software?

b) Realizar el proceso de búsqueda.

Como parte del protocolo para buscar estudios primarios, se identificaron las fuentes de información y se definió la cadena de búsqueda, según la pregunta de investigación principal. Las siguientes bases de datos se utilizaron como fuentes de información: ACM *Digital Library*, IEEE Xplore y Springer. En el momento en que se desarrolló la revisión sistemática, se contaba con

acceso a las fuentes de información descritas anteriormente. Por esta razón, se tomó la decisión de trabajar con ellas, con la pretensión de identificar los principales métodos utilizados en la realización de pruebas desde el desarrollo ágil de software.

La cadena general creada para establecer los criterios de búsqueda se configura a partir de los datos mostrados en la tabla VIII.

TABLA VIII. CRITERIOS DE BÚSQUEDA.

Concepto	Palabras relacionadas
Method	*Practice OR Framework OR Procedure*
Test	*Testing*
Agile Software Development	*Scrum OR XP*

Fuente: esta investigación.

La cadena de búsqueda fue elaborada construyendo expresiones que utilizan los operadores booleanos *OR* y *AND*. El operador *OR* se utilizó para incorporar sinónimos del concepto de búsqueda, mientras que el operador *AND* permite agregar las palabras relacionadas en la cadena de búsqueda.

La cadena de búsqueda fue perfeccionada y equilibrada de forma iterativa, de acuerdo con las características de cada motor de búsqueda que ofrece el repositorio. Los resultados de las cadenas se pueden observar en la TABLA IX. La exploración se realizó en el periodo comprendido entre los años 2011 a 2021.

Para la ejecución de las búsquedas se examinó: título, resumen y palabras clave, dentro de los resultados obtenidos a través del uso de los motores de búsqueda de cada fuente de datos seleccionada.

TABLA IX. CADENAS DE BÚSQUEDA.

Repositorio	Cadena de búsqueda	No. Artículos encontrados
ACM Digital Library	*"Method" OR "Practice" AND "Test" AND "Agile Software Development"*	37
IEEE Xplore	*"Method" AND "Test" OR "Testing" AND "Agile Software Development"*	33
Springer	*"Method" AND "Test" AND "Agile Software Development"*	74

Fuente: esta investigación.

c) Seleccionar los trabajos relevantes.

Después de realizar el proceso de búsqueda, se procedió a seleccionar los estudios relevantes, es decir, aquellos artículos que permiten dar respuesta a las preguntas de investigación planteadas. Para determinar la relevancia de los estudios, se establecieron unos criterios de inclusión y exclusión, como se puede observar en la tabla X.

Para la aplicación de los criterios de inclusión y exclusión, como parte de la selección, se definieron los filtros (F) que se muestran en la tabla XI.

TABLA X. CRITERIOS DE SELECCIÓN DE ESTUDIOS.

Criterios de inclusión	Criterios de exclusión
a. Artículos que se relacionan con los métodos utilizados en la realización de pruebas desde el desarrollo ágil de software	a. Artículos que no se relacionen con métodos utilizados en la realización de pruebas desde el desarrollo ágil de software.
b. Artículos publicados en una ventana de observación entre los años 2011 y 2021	b. Artículos publicados en una ventana de observación menor al año 2011
c. Artículos escritos en inglés.	c. Estudios duplicados.
d. Artículos, resultado de estudios primarios.	d. Reportes técnicos.

TABLA XI. ESTRATEGIA DE SELECCIÓN.

Filtro	Descripción	Criterio aplicado	
		Inclusión	Exclusión
F1	Buscar los artículos aplicando la cadena de búsqueda en los motores de las fuentes seleccionadas.	a	a
F2	Identificar artículos con la fecha dentro del rango de búsqueda.	b	b
F3	Leer el título, palabras clave y resumen del artículo aplicando los criterios de inclusión y exclusión.	a y c	a y c
F4	Leer los resultados y conclusiones del artículo aplicando los criterios de inclusión y exclusión.	a y d	a y d

Fuente: esta investigación.

En la Fig. 11, se observa la cantidad de artículos que superan los filtros de selección y aquellos que fueron descartados.

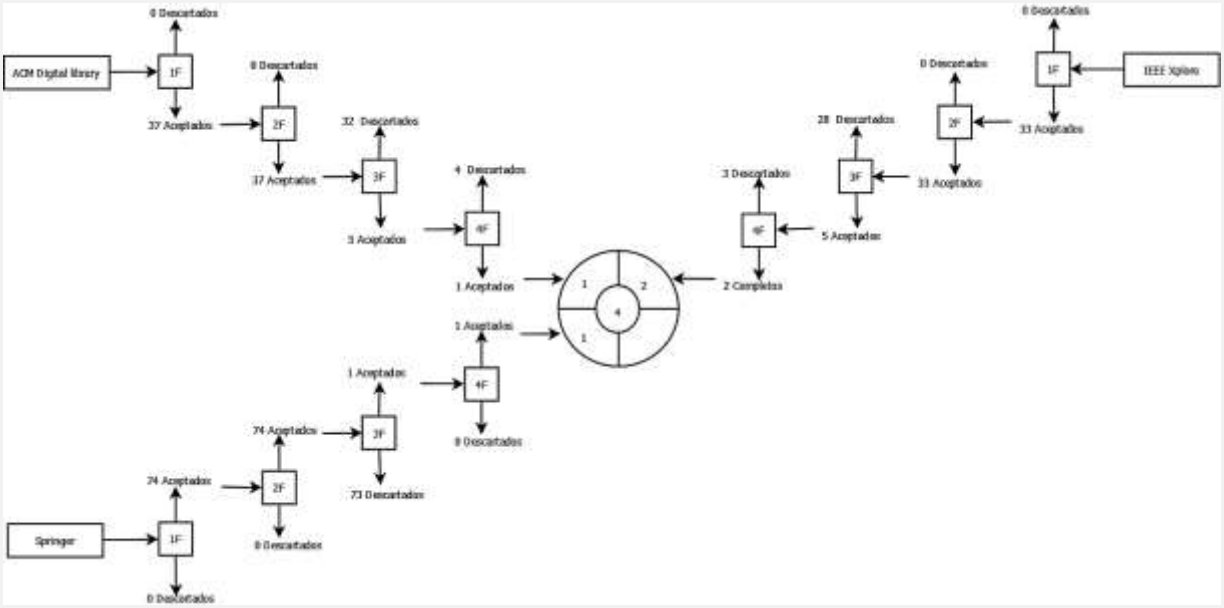


Fig. 11 Resultados de la revisión sistemática.

Fuente: esta investigación.

En la tabla XII se muestran los artículos que superan los criterios de selección definidos en la tabla X.

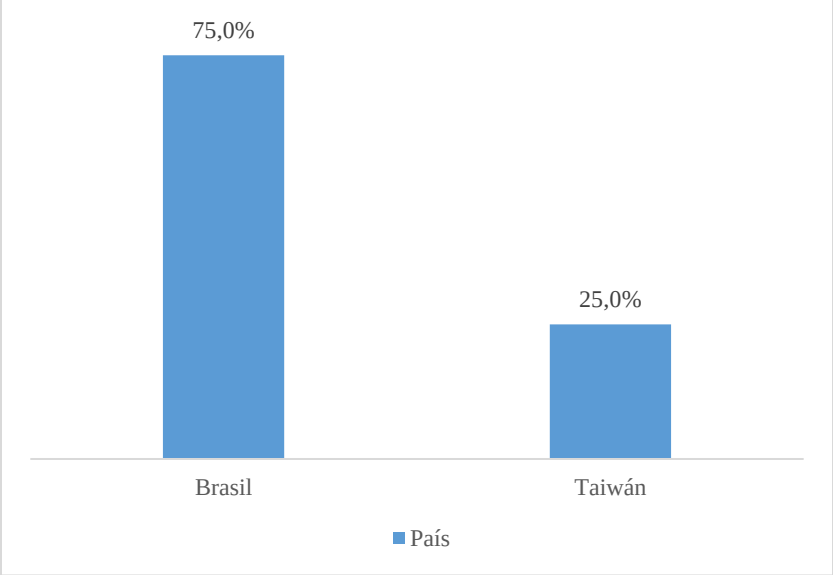
TABLA XII. ARTÍCULOS QUE SUPERARON LOS CRITERIOS DE SELECCIÓN.

ID	Artículo
A01	*AGILUS: A Method for Integrating Usability Evaluations on Agile Software Development.*
A02	*An Industrial Experience on the Application of Distributed Testing in an Agile Software Development Environment.*
A03	*A Version Control-Based Continuous Testing Frame for Improving the IID Process Efficiency and Quality.*
A04	*Software test automation practices in agile development environment: an industry experience report.*

Fuente: esta investigación.

d) Extraer datos de los artículos.

El primer aspecto general identificado, fue la ubicación geográfica de los autores de los documentos revisados con el fin de establecer porcentualmente la procedencia de los artículos por países. El porcentaje de producción para cada país fue calculado con base en la afiliación de los autores de cada documento. En la Gráfica 24 se puede observar que el país con mayor número de publicaciones relacionadas con pruebas automáticas desde ASD (*Agile Software Development* por sus siglas en inglés) es Brasil, con una frecuencia del 75,0% (equivalente a tres artículos).



Gráfica 24. País de afiliación del artículo.

Fuente: esta investigación.

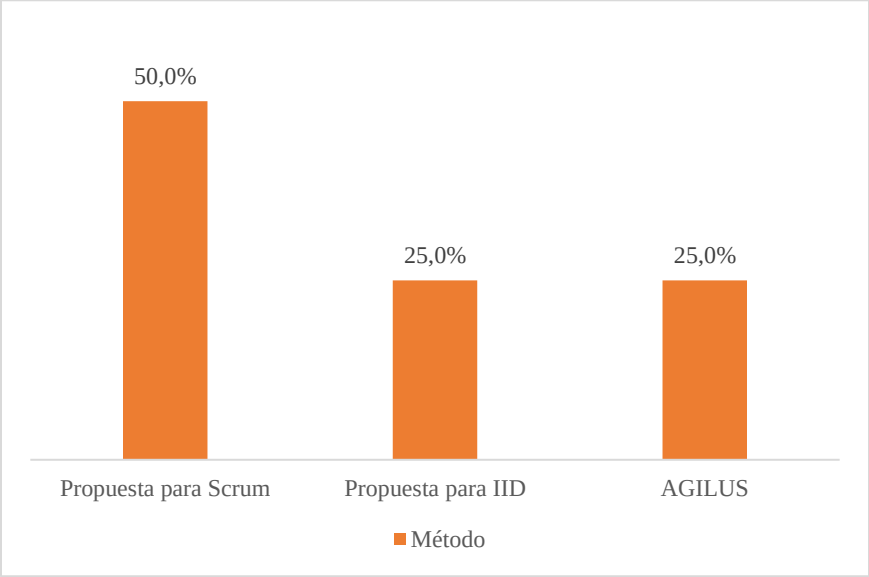
Al observar el año de publicación de los artículos, se encontró que el 75,0% (equivalente a tres artículos) fueron publicados en el año 2016, mientras que el 25,0% (equivalente a un proyecto) se publicó en el año 2012, como se muestra en la línea de tiempo que se presenta a continuación.

Fig. 12. Años de publicación.

Fuente: esta investigación.

En la Gráfica 25, se pueden observar los métodos de desarrollo ágil descritos en los artículos, que fueron utilizados para la aplicación de pruebas.

Se identifican el método AGILUS y las propuestas para IID (Desarrollo incremental iterativo) y Scrum, de los cuales, el 50,0% (equivalente a dos artículos) emplean la propuesta de aplicación de pruebas para el método Scrum, en comparación con AGILUS e IID que se emplean con una frecuencia del 25,0% (equivalente a un artículo).

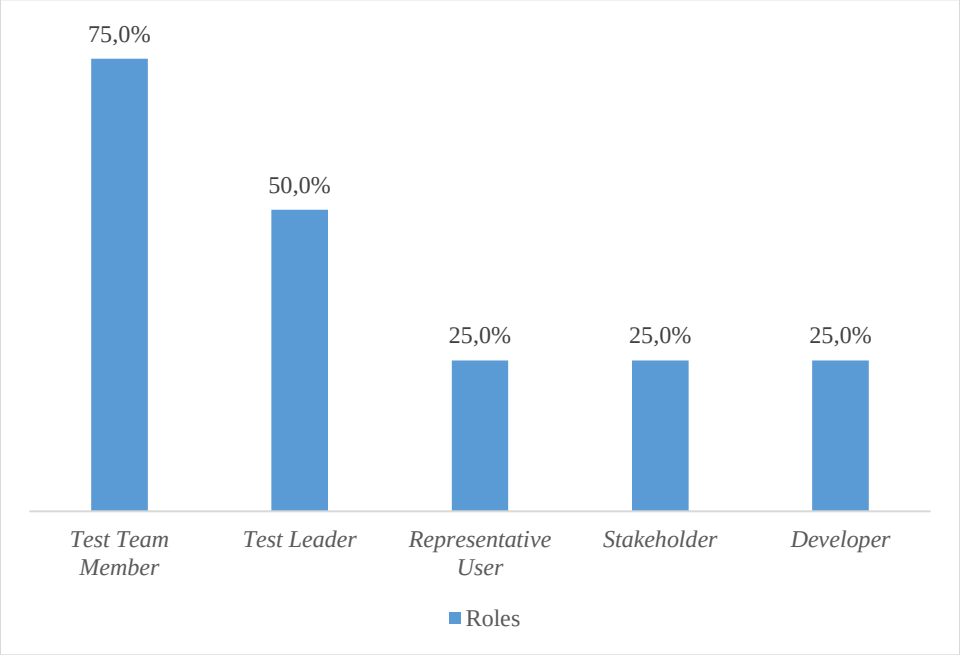


Gráfica 25. Método ágil para el desarrollo de pruebas automáticas.

Fuente: esta investigación.

Analizando el papel que desempeñan las personas que aplican las pruebas, acorde con los métodos encontrados en los artículos, se identificaron cinco roles: *Developer, Representative User, Stakeholder, Test Leader, y Test Team Member*.

La Gráfica 26, muestra que el *Test Team Member* lidera en la presencia de los artículos con una frecuencia del 75,0% (equivalente a tres artículos).



Gráfica 26. Roles que intervienen en los métodos identificados para el desarrollo de pruebas.

Fuente: esta investigación.

El método de aplicación de pruebas para AGILUS se registró en las tablas XIII, XIV, XV y XVI, en las que se encuentran las fases, actores, actividades por fase y flujos de control, extraídos de [28]

TABLA XIII. FASES DEL MÉTODO AGILUS.

ID	Nombre de la fase	Descripción
F1	Diseñar la interacción	Apoyo en la concepción del proyecto y validación del diseño de la interfaz
F2	Verificar la implementación	Verificación de los requisitos implementados en la iteración
F3	Validar el incremento del producto	Validación de incrementos de software

Fuente: esta investigación.

TABLA XIV. ACTORES DEL MÉTODO AGILUS.

ID	Nombre del actor	Descripción
AC1	*Stakeholder*	Interesados en el producto software
AC2	*Developer*	Responsable de la codificación del producto software
AC3	*Representative User*	Usuarios encargados de interactuar con el producto software

Fuente: esta investigación.

TABLA XV. ACTIVIDADES DE LAS FASES DEL MÉTODO AGILUS.

ID	Fase	Nombre de la actividad	Descripción
ACT1	F1	Diseñar prototipo de interfaz	Elaboración de vistas del software
ACT2	F1	Evaluar recorrido pluralista en el prototipo	Responsable de la validación del diseño de los prototipos de interfaz
ACT3	F1	Ajustar prototipo de interfaz	Realizar modificaciones a las vistas del software
ACT4	F2	Iniciar Sprint	Empezar con el desarrollo del Sprint
ACT5	F1	Modificar alcance de la implementación	Ajustes en la implementación para alcanzar principios de usabilidad
ACT6	F2	Efectuar prueba analítica y exploratoria	Verifica los incrementos efectuados en el software
ACT7	F2	Realizar evaluación heurística	Comprobar los principios de usabilidad del software
ACT8	F2	Incrementar el producto	Avance en el desarrollo del producto software
ACT9	F3	Aplicar pruebas con usuarios	Interacción entre usuarios y el software
ACT10	F3	Medir la calidad de usabilidad	Evaluar la eficiencia, efectividad y satisfacción del usuario respecto al producto software
ACT11	F2	Realizar ajustes al incremento del producto	Modificar código del incremento del producto

Fuente: esta investigación.

TABLA XVI. FLUJO DE CONTROL DEL MÉTODO AGILUS.

ID	Descripción	Decisión
CF1	El diseño de prototipos de interfaz se verifica desde diferentes puntos de vista	Si: ACT4 No: ACT3
CF2	Se cumplen los principios de usabilidad en los incrementos del producto	Si: ACT8 No: ACT5
CF3	Se cumple la calidad de usabilidad	Si: Fin No: ACT11

Fuente: esta investigación.

Para el método de aplicación de pruebas de Scrum, en las tablas XVII, XVIII, XIX y XX se puede observar las fases, actores, actividades y flujos de control, según [29]

TABLA XVII. FASES DE LA PROPUESTA DE APLICACIÓN DE PRUEBAS PARA SCRUM.

ID	Nombre de la fase	Descripción
F1	Ejecutar pruebas incrementales	Se realizan pruebas exploratorias y automatización de pruebas de regresión.

Fuente: esta investigación.

TABLA XVIII. ACTORES DE LA PROPUESTA DE APLICACIÓN DE PRUEBAS PARA SCRUM

ID	Nombre del actor	Descripción
AC1	*Test Leader*	Líder del equipo de pruebas
AC2	*Test Team Member*	Valida la calidad del producto software

Fuente: esta investigación.

TABLA XIX. ACTIVIDADES DE LAS FASES DE LA PROPUESTA DE APLICACIÓN DE PRUEBAS PARA SCRUM.

ID	Fase	Nombre de la actividad	Descripción
ACT1	F1	Configurar ambiente de pruebas	Preparar ambiente de desarrollo de pruebas
ACT2	F1	Planear actividades de pruebas	Diseño de las actividades de prueba
ACT3	F1	Crear o actualizar casos de prueba	Construcción/actualización de casos de prueba
ACT4	F1	Revisar casos de prueba	Verificación de los casos de prueba creados
ACT5	F1	Crear o actualizar scripts de pruebas automáticas	Construcción/actualización de archivos de pruebas automáticas
ACT6	F1	Realizar ejecución incremental de pruebas	Aplicación de pruebas de software
ACT7	F1	Reportar o actualizar fallas	Informar/actualizar documento de registro de fallas
ACT8	F1	Corregir fallas	Se corrigen las fallas encontradas en el software
ACT9	F1	Generar reporte de pruebas	Obtener reporte de pruebas

Fuente: esta investigación.

TABLA XX. FLUJOS DE CONTROL DE LA PROPUESTA DE APLICACIÓN DE PRUEBAS PARA SCRUM.

ID	Descripción	Decisión
CF1	Se aprueba el caso de prueba	Si: ACT5 No: ACT3
CF2	Fallas verificadas	Si: ACT7 No: CF3
CF3	Finaliza la iteración	Si: ACT9 No: ACT3
CF4	Finaliza las historias del Backlog	Si: Fin No: ACT2

Fuente: esta investigación.

Para el método IID, en las tablas XXI, XXII, XXIII, y XXIV se puede observar las fases, actores, actividades y flujos de control extraídos de [30]

TABLA XXI. FASES DE LA PROPUESTA DE APLICACIÓN DE PRUEBAS PARA IID.

ID	Nombre de la fase	Descripción
F1	Planear casos de prueba	Diseñar y crear los casos de prueba
F2	Gestionar casos de prueba	Ejecutar y controlar los casos de prueba
F3	Revisar casos de prueba	Inspección o ajuste de los casos de prueba, realizando control de versiones
F4	Identificar errores	Encontrar fallas en el software mediante la aplicación de casos de prueba

Fuente: esta investigación.

TABLA XXII. ACTORES DE LA PROPUESTA DE APLICACIÓN DE PRUEBAS PARA IID.

ID	Nombre del actor	Descripción
AC1	*Test Team Member*	Valida la calidad del producto software

Fuente: esta investigación.

TABLA XXIII. ACTIVIDADES DE LAS FASES DE LA PROPUESTA DE APLICACIÓN DE PRUEBAS PARA IID.

ID	Fase	Nombre de la actividad	Descripción
ACT1	F1	Diseñar casos de prueba	Elaboración de casos de prueba en base a los requerimientos del software
ACT2	F1	Crear o ajustar casos de prueba	Construcción de casos de prueba para evaluación del software, o modificación de los casos de prueba
ACT3	F2	Empaquetar la información de los casos de prueba en un documento.	Registrar los casos de prueba utilizando un formato formal
ACT4	F3	Revisar casos de prueba	Inspeccionar y adecuar el caso de prueba
ACT5	F4	Ejecutar casos de prueba	Aplicar los casos de prueba en el software
ACT6	F4	Establecer errores a partir de los casos de prueba	Identificar fallas en el software a partir de los casos de prueba
ACT7	F4	Generar reporte de las pruebas	Mostrar los resultados de la ejecución de los casos de prueba

Fuente: esta investigación.

TABLA XXIV. FLUJO DE CONTROL DE LA PROPUESTA DE APLICACIÓN DE PRUEBAS PARA IID.

ID	Descripción	Decisión
CF1	El diseño de los casos de prueba es completo	Si: ACT2 No: ACT1
CF2	Los casos de prueba son válidos	Si: ACT3 No: ACT4

Fuente: esta investigación.

Con base en la información descrita en las tablas anteriores, se elaboró el diagrama BPMN por cada uno de los métodos mencionados, realizados con la herramienta *Bizagi Modeler*, como se observa en las Figuras (Fig.)13, 14 y 15

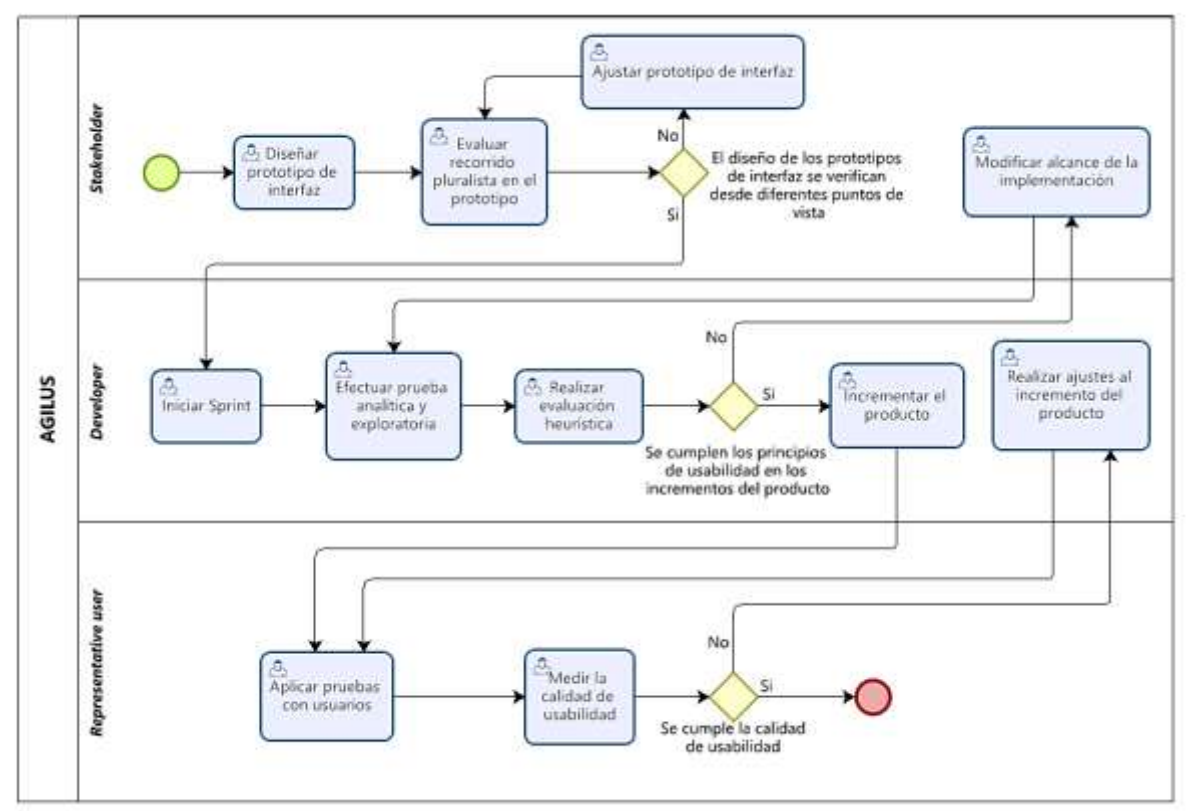


Fig. 13. Diagrama BPMN de AGILUS.

Fuente: esta investigación.

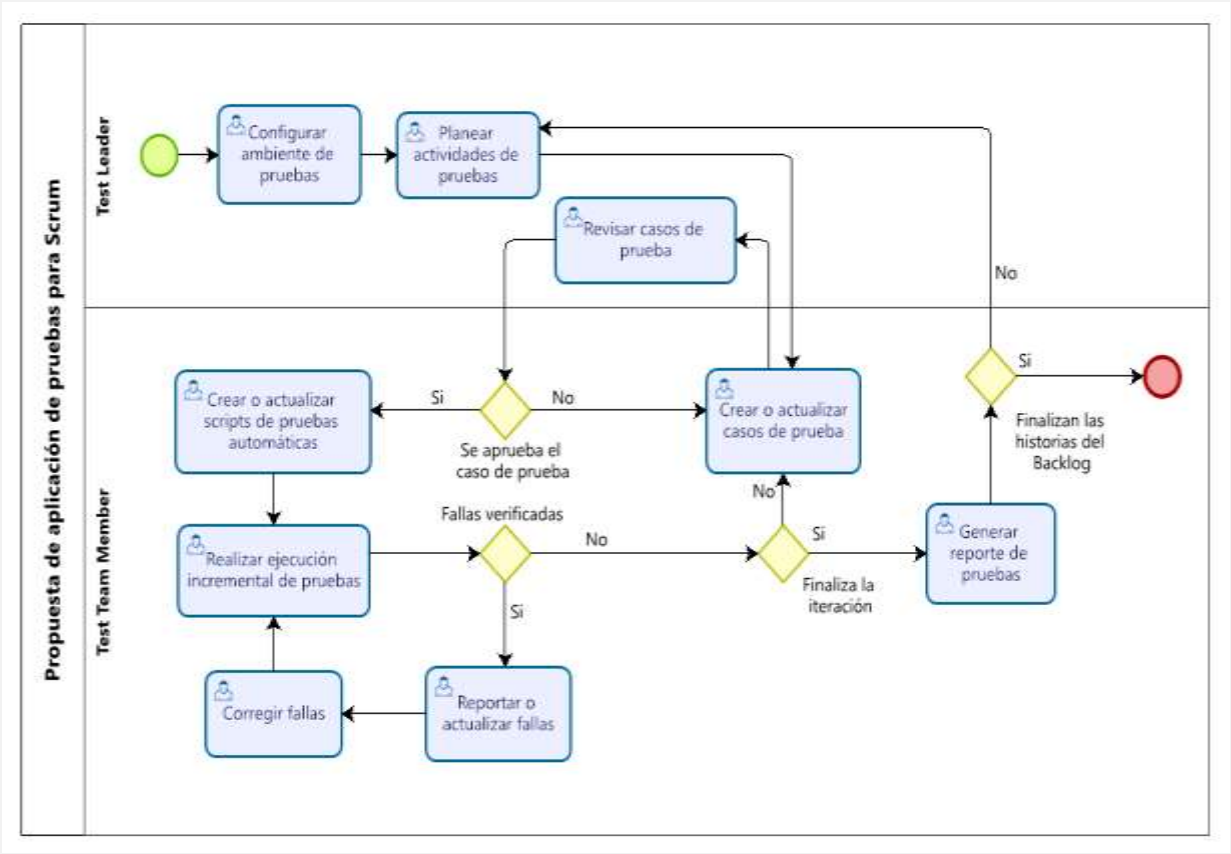


Fig. 14. Diagrama BPMN de la propuesta de aplicación de pruebas para Scrum (PPS).

Fuente: esta investigación.

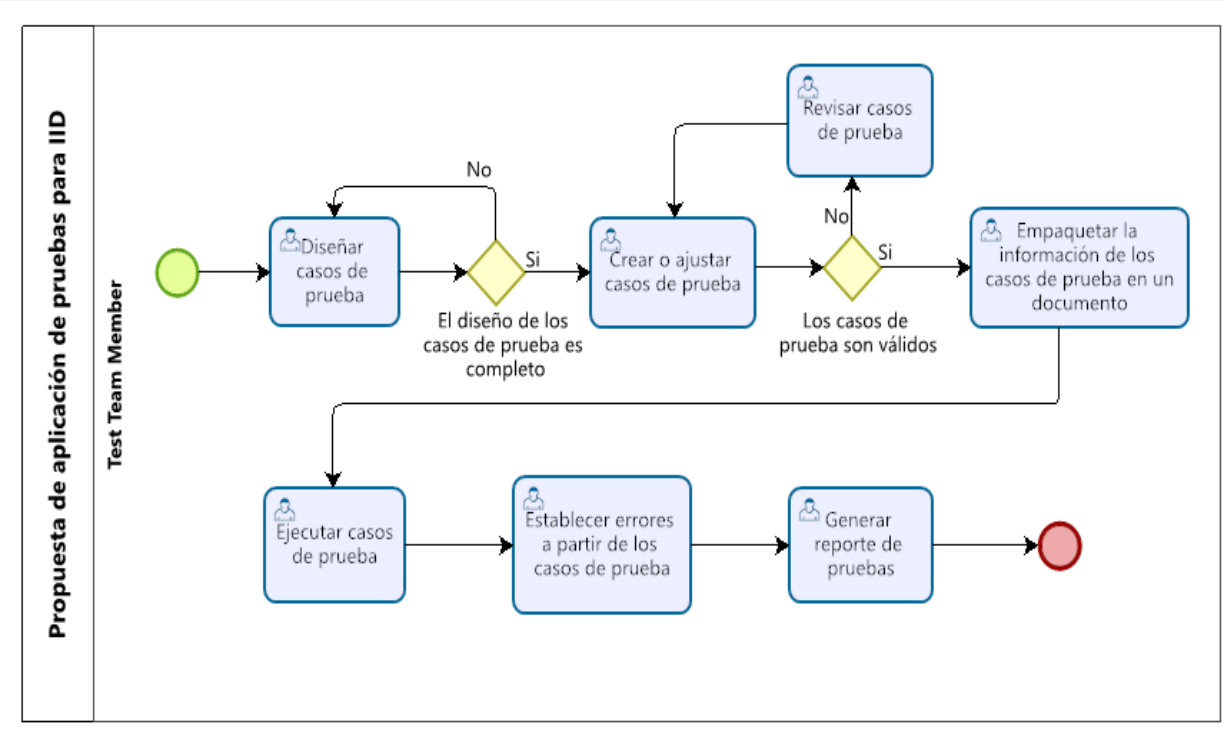


Fig. 15. Diagrama BPMN de la propuesta de aplicación de pruebas para IID (PIID).

Fuente: esta investigación.

2) Análisis comparativo de métodos para la aplicación de pruebas automáticas

En las tablas XXV, XXVI y XXVII se puede observar los aspectos positivos y por mejorar de del método AGILUS y las propuestas de aplicación de pruebas para Scrum e IID.

TABLA XXV. ASPECTOS POSITIVOS Y POR MEJORAR DEL MÉTODO AGILUS.

Método	Aspectos positivos	Aspectos por mejorar
AGILUS	Hace uso de Scrum para realizar la ejecución de pruebas. Según [42] durante el año 2021, Scrum es la más utilizada por los equipos de desarrollo con un 81,0% *(Método - Adopción)*	Se enfoca únicamente en la realización de pruebas de usabilidad. *(Método)*
	No incorpora nuevos roles para realizar el proceso de aplicación de pruebas. *(Adopción-Tiempo-Costo)*	No se evidencia que las actividades se soportan mediante el uso de tecnologías para automatizar las pruebas. *(Recurso tecnológico)*
	Se puede reducir la inversión en la adopción, ya que se utiliza un método conocido y de fácil acceso a su información. *(Costo)*	
	El *stakeholder* y *representative user* intervienen en la fase de evaluación, por tal razón, el producto software final cumple con las expectativas de dichos roles. *(Método)*	

Fuente: esta investigación.

TABLA XXVI. ASPECTOS POSITIVOS Y POR MEJORAR DE LA PROPUESTA PARA SCRUM.

Método	Aspectos positivos	Aspectos por mejorar
Propuesta de aplicación de pruebas para Scrum. (PPS)	Hace uso de Scrum para realizar la ejecución de pruebas. Según [42] durante el año 2021, Scrum es la más utilizada por los equipos de desarrollo con un 81,0% *(Método - Adopción)*	El tiempo de ejecución de Scrum podría incrementar debido a la inclusión de nuevos roles. *(Tiempo)*
	Una propuesta para Scrum requiere únicamente la incorporación de las nuevas actividades y la definición de los nuevos roles. *(Adopción)*	La inclusión de dos nuevos roles en Scrum, aumenta el costo de ejecución del proceso en Scrum. *(Costo)*
	Por ser una propuesta para Scrum, se requiere de un menor periodo para adoptarla, debido a que Scrum es reconocido, y la capacitación estaría enfocada en los nuevos roles. *(Tiempo)*	La contratación de un perfil para el *Test Leader* no es necesario, ya que en Scrum los miembros del equipo son auto gestionados, por lo cual, el *Test Team Member* está en la capacidad de cumplir las funciones del perfil mencionado. *(Costo)*
	Se puede reducir la inversión en la adopción, ya que se utiliza un método conocido y de fácil acceso a su información. *(Costo)*	
	Se evidencia que las actividades se soportan mediante el uso de tecnologías para automatizar las pruebas. *(Recurso tecnológico)*	
	Es interesante contar con el rol de un *Test Leader*, ya que cuenta con experiencia en el desarrollo y ejecución de pruebas de software. (Factor humano)	

Fuente: esta investigación.

TABLA XXVII. ASPECTOS POSITIVOS Y POR MEJORAR DE LA PROPUESTA DE APLICACIÓN DE PRUEBAS PARA IID.

Método	Aspectos positivos	Aspectos por mejorar
Propuesta de aplicación de pruebas para IID (PIID)	Realiza pruebas continuas que permiten identificar oportunamente los defectos de calidad y reducir efectivamente el riesgo en el desarrollo de software. *(Aseguramiento de la calidad)*	Utiliza el proceso IID (*Interative Incremental Development*) que es un método poco conocido en la realización de proyectos de desarrollo de software. *(Método–Adopción)*
	Incorpora el marco de prueba continuo, basado en el control de versiones (VCCTF), con el cual se reduce el tiempo y factor humano. *(Tiempo–Costo-Factor humano)*	Se requiere de tiempo para capacitar al personal en relación con el método propuesto en IID. *(Tiempo)*
	Incorpora el marco de prueba continuo, basado en el control de versiones (VCCTF), que permite la implementación e integración de nuevas versiones. *(Método)*	
	La fase de pruebas es realizada por un único rol. *(Costo)*	
	Se evidencia que las actividades se soportan mediante el uso de tecnologías para automatizar las pruebas. *(Recurso tecnológico)*	

Fuente: esta investigación.

En la TABLA XXVIII se presenta los aspectos positivos de los métodos para aplicación de pruebas, de acuerdo con cada una de las categorías identificadas: método, adopción, costo, tiempo, recurso tecnológico, factor humano y aseguramiento de calidad. Se resalta lo común en cada una de las categorías, y en el campo observación se describe la coincidencia entre los métodos.

TABLA XXVIII. ASPECTOS POSITIVOS DE LOS MÉTODOS PARA APLICACIÓN DE PRUEBAS.

Categoría	AGILUS	Propuesta para Scrum	Propuesta para IID	Observación
Método	Hace uso de Scrum para realizar la ejecución de pruebas. Según [42] durante el año 2021, Scrum es la más utilizada por los equipos de desarrollo con un 81,0%	Hace uso de Scrum para realizar la ejecución de pruebas. Según [42] durante el año 2021, Scrum es la más utilizada por los equipos de desarrollo con un 81,0%	Incorporar el marco de prueba continuo, basado en el control de versiones (VCCTF), permite la implementación e integración de nuevas versiones.	Uso de Scrum
	El stakeholder y representative user intervienen en la fase de evaluación, por tal razón, el producto software final cumple con las expectativas de dichos roles.		El usuario puede asistir e involucrarse en el desarrollo del proyecto software.	El rol de usuario interviene en el proyecto software
Adopción	No incorpora nuevos roles para realizar el proceso de aplicación de pruebas.	Una propuesta para Scrum requiere únicamente la incorporación de las nuevas actividades y la definición de los nuevos roles.		
	Hace uso de Scrum para realizar la ejecución de pruebas. Según [42] durante el año 2021, Scrum es la más utilizada por los equipos de desarrollo con un 81,0%	Hace uso de Scrum para realizar la ejecución de pruebas. Según [42] durante el año 2021, Scrum es la más utilizada por los equipos de desarrollo con un 81,0%		Emplear un proceso conocido.
Costo	No incorpora nuevos roles para realizar el proceso de aplicación de pruebas.		La fase de pruebas es realizada por un único rol.	
	Se puede reducir la inversión en la adopción, ya que se utiliza un método conocido y de fácil acceso a su información.	Se puede reducir la inversión en la adopción, ya que se utiliza un método conocido y de fácil acceso a su información	Incorpora el marco de prueba continuo, basado en el control de versiones (VCCTF), con el cual se reduce el tiempo y factor humano.	Reducción de inversión por método conocido.
Tiempo	No incorpora nuevos roles para realizar el proceso de aplicación de pruebas.	Por ser una propuesta para Scrum, se requiere de un menor periodo para adoptarla, debido a que Scrum es reconocido, y la capacitación estaría enfocada en los nuevos roles.	Incorpora el marco de prueba continuo, basado en el control de versiones (VCCTF), con el cual se reduce el tiempo y factor humano.	

Categoría	AGILUS	Propuesta para Scrum	Propuesta para IID	Observación
Recurso tecnológico		Se evidencia que las actividades se soportan mediante el uso de tecnologías para automatizar las pruebas.	Se evidencia que las actividades se soportan mediante el uso de tecnologías para automatizar las pruebas.	Uso de herramientas para automatización de pruebas.
Factor humano		Es interesante contar con el rol de un Test Leader, ya que cuenta con experiencia en el desarrollo y ejecución de pruebas de software.	Incorpora el marco de prueba continuo, basado en el control de versiones (VCCTF), con el cual se reduce el tiempo y factor humano.	
Aseguramiento de calidad			Realiza pruebas continuas que permiten identificar oportunamente los defectos de calidad y reducir efectivamente el riesgo en el desarrollo de software.	

Fuente: esta investigación.

A continuación, en la tabla XXIX se listan los aspectos por mejorar identificados en cada uno de los métodos. Se resalta lo común en las categorías, y en el campo observación se describe la propuesta para mejorar los métodos.

TABLA XXIX. ASPECTOS POR MEJORAR DE LOS MÉTODOS PARA APLICACIÓN DE PRUEBAS

Categoría	AGILUS	Propuesta para Scrum	Propuesta para IID	Observación
Método	Se enfoca únicamente en la realización de pruebas de usabilidad.		Utiliza el proceso IID (Interative Incremental Development) que es un método poco conocido en la realización de proyectos de desarrollo de software.	La propuesta se debe enfocar en pruebas automáticas de unidad y componentes.
Adopción			Utiliza el proceso IID (Interative Incremental Development) que es un método poco conocido en la realización de proyectos de desarrollo de software.	Emplear un proceso conocido.
Costo		La inclusión de dos nuevos roles en el método, aumenta el costo de ejecución del proceso en Scrum. La contratación de un perfil para el Test Leader no es necesario, ya que en Scrum los miembros del equipo son auto gestionados, por lo cual, el Test Team Member está en la capacidad de cumplir las funciones del perfil mencionado.		Explorar la posibilidad de manejar más de un rol para pruebas. Hacer uso de técnicas para fomentar la autogestión de los roles.
Tiempo		El tiempo de ejecución de Scrum podría incrementar debido a la inclusión de nuevos roles.	Se requiere de tiempo para capacitar al personal en relación con el método propuesto en IID.	Elaborar una propuesta que mantenga los valores y principios propuestos en el manifiesto por el desarrollo ágil de software.
Recurso tecnológico	No se evidencia que las actividades se soportan mediante el uso de tecnologías para automatizar las pruebas.			La propuesta debe hacer uso de herramientas de automatización para pruebas de unidad y componentes.

Fuente: esta investigación.

3) Elementos de la propuesta Success Process UCT

En esta sección se sintetizan los elementos a incluir dentro del proceso que se propone para realizar pruebas automáticas de unidad y componentes, a partir del análisis comparativo realizado al método AGILUS y las propuestas de aplicación de pruebas para Scrum (PPS) e IID (PIID).
En la Fig. 16, se muestran los aspectos tomados de los métodos en color gris, colocando a un lado el método del que fue extraído y aquellos planteados en color verde, que conforman la propuesta *Success Process UCT (Unit And Component Tests* por sus siglas en inglés*).*

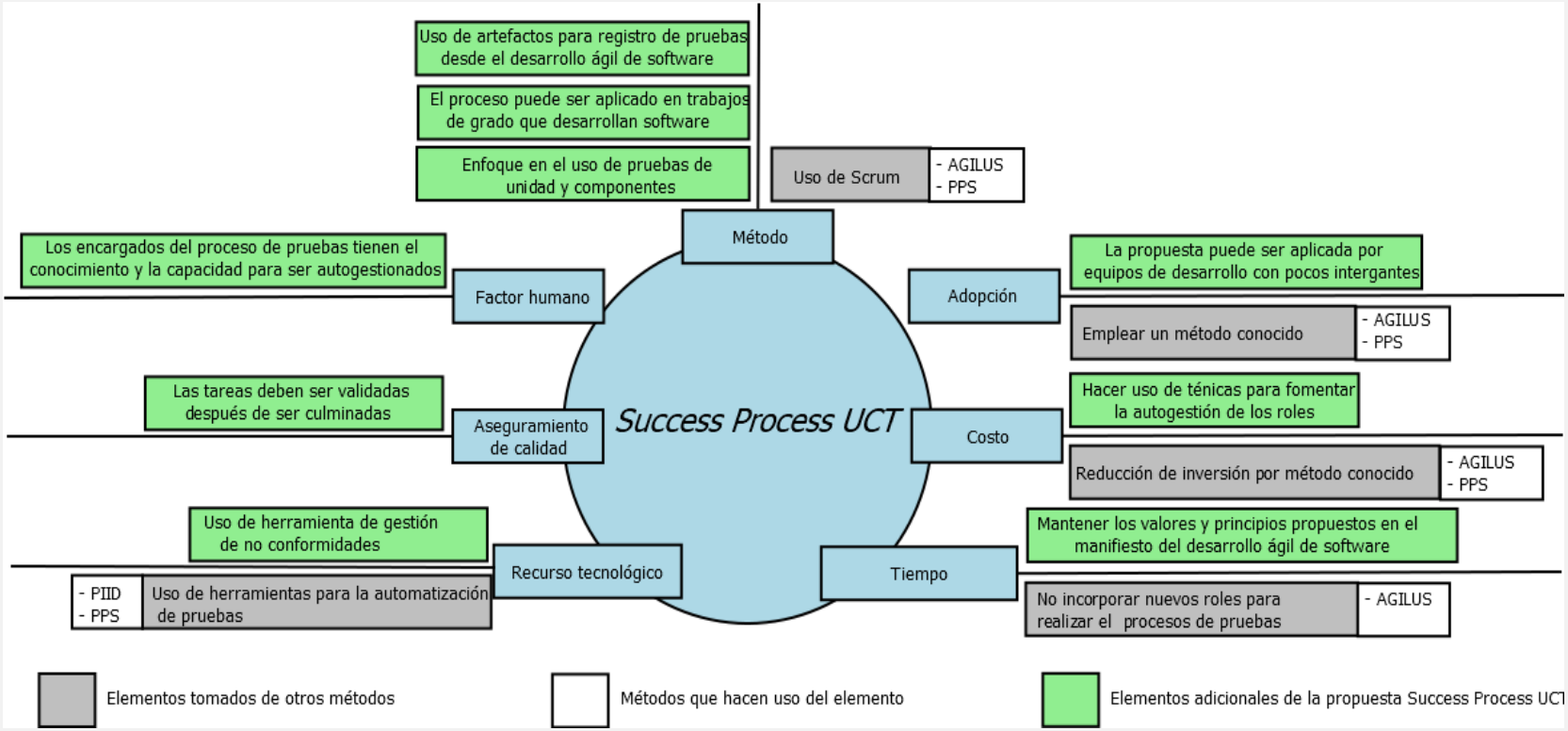


Fig. 16. Esquema de la propuesta *Success Process UCT*.

Fuente: esta investigación.

a) Decisión en relación con Scrum para pruebas automáticas de unidad y componentes.

Prácticas de aseguramiento de la calidad en Scrum para pruebas automáticas de unidad y componentes.

Con el fin de no afectar los principios de desarrollo ágil, se plantea una solución que respeta los roles definidos en el método seleccionado (Scrum); por lo tanto, se pretende que los actores ya existentes adquieran las capacidades y conocimientos necesarios, para realizar las actividades de pruebas automáticas de unidad y componentes; además, con estos nuevos conocimientos en cada rol se fomentará prácticas de autogestión.

Entre las ventajas de la propuesta se tiene: implementación en equipos de desarrollo con pocos integrantes, la posibilidad de ser aplicado en proyectos de grado, ya que se enfoca en las prácticas a realizar en la fase de pruebas y no en el rol que las desempeñaría, por lo cual es factible que cualquier miembro del equipo pueda realizarla conforme con sus necesidades, adicionalmente se asegura la calidad del software, verificando cada actividad una vez sea culminada.

En la tabla XXX se puede observar las decisiones tomadas para elaborar la propuesta *Success Process UCT.*

TABLA XXX. DECISIONES PARA LA CREACIÓN DE LA PROPUESTA *SUCCESSS PROCESS UCT.*

Id	Decisión	Categoría
D1	Usar Scrum, así como artefactos que permitan al *Product Owner* registrar las pruebas desde el desarrollo ágil de software; además, un integrante del *Development Team* requiere realizar tareas para la ejecución de las pruebas, por lo tanto se necesita definir las prácticas a efectuar.	Método, Adopción
D2	Las prácticas deben estar basadas en técnicas que permitan fomentar la autogestión.	Costo
D3	Las prácticas deben ser ágiles.	Tiempo
D4	Las tareas deben ser inspeccionadas al estar en estado *Done*, por lo tanto se debe definir prácticas ágiles.	Aseguramiento de la calidad
D5	Usar las herramientas para realizar pruebas unitarias y de componentes, así como herramientas de gestión de no conformidades.	Recurso técnológico
D6	Definir un proceso de capacitación para el equipo en relación con el desarrollo pruebas automáticas unitarias y de componentes.	Factor Humano

Fuente: esta investigación.

Las restricciones a considerar en el método propuesto se muestran en LA TABLA XXXI.

TABLA XXXI. RESTRICCIONES PARA LA PROPUESTA *SUCCESS PROCESS UCT*.

Id	Descripción
R1	Los trabajos de grado se desarrollan entre 2 y 3 estudiantes.
R2	Tener conocimientos del método Scrum.

Fuente: esta investigación.

Para la propuesta *Success Process UCT*, en las tablas XXXII, XXXIII, XXXIV y XXXV se presentan las fases, actores, actividades y flujos de control planteados.

TABLA XXXII. FASES DE LA PROPUESTA *SUCCESS PROCESS UCT*.

Id	Nombre de la fase	Descripción
F1	Planeación y preparación de pruebas	Especificar, identificar escenarios de pruebas de unidad y componentes, diseñar casos de prueba y configurar el ambiente de acuerdo con el lenguaje de codificación.
F2	Codificación de casos de prueba	Crear los casos de prueba y verificar que la HU esté codificada.
F3	Aplicación/verificación de pruebas	Ejecutar casos de prueba, generación y consulta de reporte de pruebas, verificación de no conformidades y modificación de escenarios de prueba.
F4	Gestión de no conformidades	Gestión de métricas en relación con las no conformidades y generación de alertas.
F5	Aseguramiento de calidad	Revisar, corregir, cambiar estado, generar notificación de corrección de no conformidades y revisión de la HU.

Fuente: esta investigación.

TABLA XXXIII. ACTORES DE LA PROPUESTA *SUCCESS PROCESS UCT*.

ID	Nombre del actor	Descripción
AC1	*Product Owner*	Maximiza el valor del producto software mediante métodos ágiles.
AC2	*Development Team Member For Testing*	Valida la calidad del producto software.
AC3	*Development Team Member*	Codifica el producto software.
AC4	Herramienta automática para pruebas unitarias	Facilita la ejecución de pruebas automáticas de unidad.
AC5	Herramienta de gestión de no conformidades	Facilita el registro de los eventos ocurridos en la aplicación de pruebas y la generación de notificaciones.

Fuente: esta investigación.

TABLA XXXIV. ACTIVIDADES DE LAS FASES DE LA PROPUESTA *SUCCESS PROCESS UCT*.

ID	Fase	Nombre de la actividad	Descripción
ACT1	F1	Especificar historias de usuario (HU)	Detallar los requerimientos de la HU
ACT2	F1	Identificar escenarios de pruebas unitarias y de componentes para la HU	Reconocer escenarios para la aplicación de pruebas de unidad y componentes
ACT3	F1	Diseñar casos de prueba	Planear los casos de prueba
ACT4	F1	Configurar el ambiente de pruebas de acuerdo con el lenguaje de codificación	Modelar ambiente para pruebas de unidad
ACT5	F2	Crear casos de prueba	Construcción de casos de prueba
ACT6	F2	Verificar que la HU esté codificada/modificada	Comprobar que la construcción o ajuste de la HU haya finalizado
ACT7	F2	Codificar HU	Construir HU
ACT8	F3	Ejecutar casos de prueba	Aplicar casos de prueba
ACT9	F3	Generar reporte con los resultados de las pruebas	Creación automática de reporte de resultados de pruebas unitarias
ACT10	F3	Consultar reporte de resultados de pruebas	Revisar reporte
ACT11	F3	Verificar no conformidades en el reporte de pruebas	Revisar no conformidades encontradas en los escenarios de pruebas
ACT12	F3	Modificar escenarios de prueba	Ajustar/actualizar los escenarios de prueba
ACT13	F5	Revisar funcionamiento de la HU	Verificar que la HU sea funcional
ACT14	F4	Registrar métricas en relación con las no conformidades en la herramienta de gestión las mismas.	Registrar no conformidades identificadas, tipo de la no conformidad, prioridad, iteración, y la HU en la que se encuentra la no conformidad
ACT15	F4	Almacenar métricas de las no conformidades	Guardar información de las métricas de las no conformidades encontradas
ACT16	F4	Generar alerta de no conformidades	Notificar al desarrollador sobre la no conformidad encontrada
ACT17	F5	Revisar no conformidades encontradas en la HU	Inspeccionar no conformidades identificadas
ACT18	F5	Corregir las no conformidades encontradas en la HU	Ajustar HU
ACT19	F5	Cambiar estado de las no conformidades en la herramienta de gestión de no conformidades	Cambiar estado de la no conformidad
AC20	F5	Generar notificación de corrección de no conformidades al *Development Team Member For Testing*	Enviar notificación de ajuste de no conformidades en la HU

Fuente: esta investigación.

TABLA XXXV. FLUJO DE CONTROL DE LA PROPUESTA *SUCCESS PROCESS UCT*.

ID	Descripción	Decisión
CF1	La HU está codificada	Si: ACT8 No: ACT7
CF2	Se encuentran no conformidades en los escenarios de prueba	Si: ACT11 No: ACT12
CF3	Se encuentran no conformidades en la codificación de la HU	Si: ACT14 No: ACT13
CF4	HU conforme	Si: Fin (HU en estado *DONE*) No: ACT1

Fuente: esta investigación.

Con base en la información descrita en las tablas anteriores, se elaboró el diagrama BPMN del proceso de la propuesta *Success Process UCT* con la herramienta *Bizagi Modeler*[5], como se observa en la Fig. 17, la cual corresponde a la versión cuatro del modelo realizado. Las versiones previas se pueden consultar en el Anexo F.

La versión final del modelo fue validada mediante el juicio de expertos, para lo cual fue socializado a un experto tester que cuenta con varios años de experiencia en el área de pruebas. En el Anexo E se sintetiza la información de la propuesta para aplicación de pruebas *Success Process UCT*.

[5] Bizagi Modeler es una marca registrada de la empresa Bizagi que se puede descargar de https://www.bizagi.com/es/plataforma/modeler

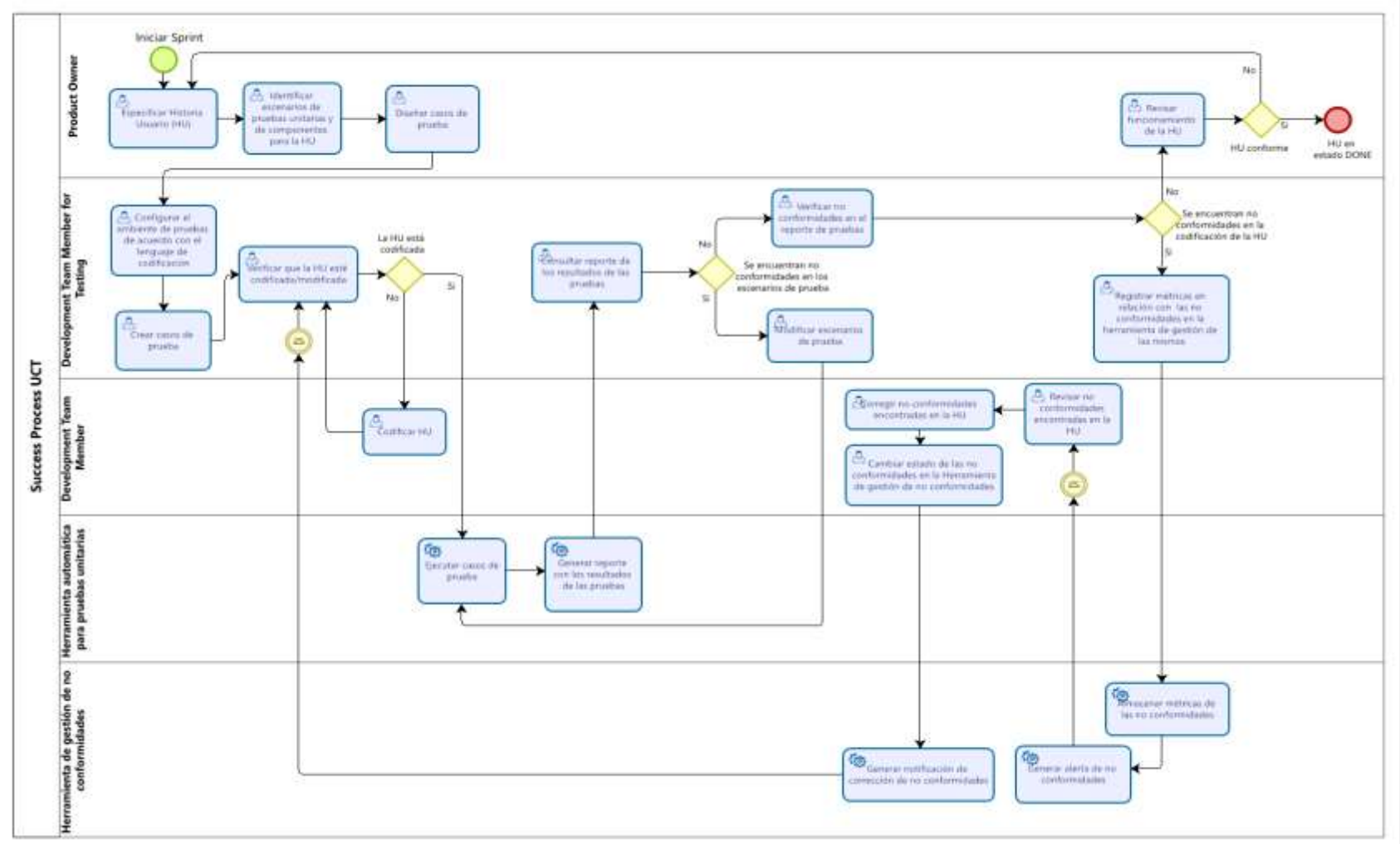


Fig. 17. Diagrama BPMN de la propuesta *Success Process UCT*.

Fuente: Esta investigación.

4) Sintetizar resultados

El referente teórico seleccionado para representar los métodos de aplicación de pruebas es la notación descrita por [22], denominada *Business Process Management Notation* (BPMN por sus siglas) y la herramienta *Bizagi Modeler* para graficarlos.

Se realiza un mapeo sistemático de métodos para la ejecución de pruebas automáticas desde el desarrollo ágil de software, en los repositorios: ACM *Digital Libray*, IEEE Xplore, y Springer. La búsqueda permitió identificar un total de 144 artículos, de los cuales se seleccionaron cuatro estudios, extrayendo los métodos: AGILUS, y las propuestas para IID y Scrum. Al inspeccionar los artículos se encontró que el método de aplicación de pruebas que lidera, es la propuesta para Scrum con el 50,0% (equivalente a dos artículos), el país con mayor afiliación es Brasil, el año en que más se publicaron fue 2016 y el rol más común en la fase de pruebas es el *Test Team Member*, con una frecuencia de 75,0% (equivalente a tres artículos).

Con base en la evaluación de los métodos se realiza la propuesta *Success Process UCT (Unit and Component Tests)* basado en las restricciones: los trabajos de grado se desarrollan entre dos a tres integrantes y se tiene conocimientos sobre el método Scrum; además, se incluye los aspectos: no incorporación de nuevos roles, uso de Scrum ya que es un método conocido y la implementación de herramientas para automatizar pruebas, aportando nuevos elementos como: la inclusión de artefactos para la especificación de historias de usuario (HU), escenarios y casos de prueba, herramientas de gestión de no conformidades, capacidad para aplicar la propuesta en trabajos de grado que desarrollan software, enfoque en pruebas de unidad y componentes, y la validación de tareas ya culminadas.

El método está conformado por las fases: planeación y preparación de pruebas, codificación de casos de prueba, aplicación/verificación de pruebas, gestión de no conformidades y aseguramiento de calidad, las cuales se desarrollan a través de un conjunto de actividades.

El *Product Owner* se encarga de especificar la HU, identificar los escenarios, diseñar las pruebas, verificar el funcionamiento de la HU y su cambio de estado a *DONE*. El *Development Team Member For Testing* es quien configura el ambiente y crea los casos de prueba, verifica la codificación de la HU, consulta el reporte de los resultados de pruebas, examina las no conformidades, registra las métricas de las mismas y en la herramienta de gestión y modifica los escenarios de prueba. El *Development Team Membe*r codifica la HU, revisa, modifica y cambia el estado de las no conformidades, la herramienta automática para pruebas unitarias le permite ejecutar los casos de prueba y generar un reporte con los resultados; y la herramienta de gestión de no conformidades almacena datos de las métricas y permite gestionar alertas.

C. VALIDACIÓN DE LA PROPUESTA SUCCES PROCESS UCT

En esta sección, se presenta el proceso de validación de la propuesta *Succes Process UCT* para el desarrollo y ejecución de pruebas automáticas de unidad y componentes desde el enfoque ASD. Las etapas realizadas en el proceso de validación se presentan en la Fig. 18.

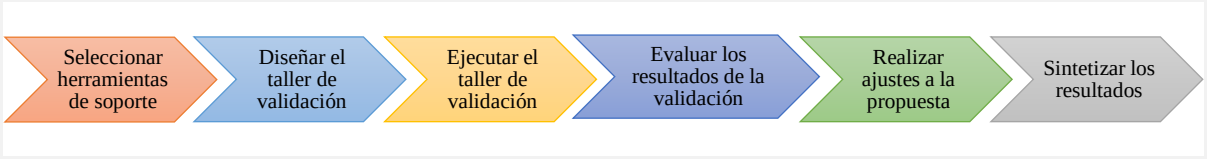


Fig. 18. Esquema de la propuesta *Success Process UCT*

Fuente: esta investigación.

1) Seleccionar herramientas de soporte

El primer paso para la validación es identificar una herramienta que según los planteamientos de [31] permita recopilar datos para generar información que posibilite mejorar el desempeño en el desarrollo de las pruebas. Para identificar las herramientas más relevantes que brindarán soporte a la propuesta *Success Process UCT*, se realizó una búsqueda sistemática de software utilizando elementos propuestos en el protocolo planteado por [32] El proceso de revisión sistemática de la propuesta se elaborará con base en el protocolo descrito en [31] y se presenta en la Fig. 19.

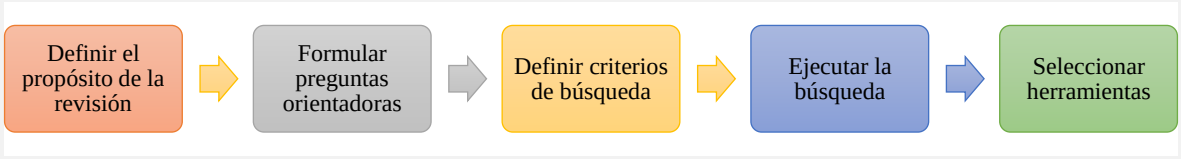


Fig. 19. Proceso de revisión sistemática de la propuesta *Success Process UCT*.

Fuente: esta investigación.

A continuación, se muestran los resultados de la realización de cada actividad del proceso.

a) Definición del propósito de la revisión

El fin de la búsqueda de herramientas software está orientado hacia la gestión de no conformidades y posibilitar el trabajo colaborativo, porque según [33] se hace necesario hacer uso de software para gestionar el desarrollo de software como una habilidad relevante en los profesionales en tecnología.

b) Formulación de las preguntas orientadoras

En este apartado, se condensan las preguntas que orientarán la revisión sistemática de software para la gestión de no conformidades encontradas en la aplicación de pruebas [31] Teniendo en cuenta estos fines, se plantea la pregunta:

RQ1. ¿Qué software permite hacer seguimiento de no conformidades encontradas en pruebas automáticas de software y trabajar de manera colaborativa?

c) Definición de los criterios de búsqueda

Como parte del protocolo para buscar el software, se definió la cadena de búsqueda según la pregunta orientadora. El repositorio *SourceForge,* se utilizó como fuente de información. La cadena general creada para establecer los criterios de búsqueda se configura a partir de los datos mostrados en la tabla XXXVI.

TABLA XXXVI. CADENA DE BÚSQUEDA.

Concepto	Palabras relacionadas
Bug Tracking	*(Error OR Failure) AND (Tracker)*

Fuente: esta investigación.

Para la ejecución de la búsqueda, se examinó: nombre de la herramienta software, año de publicación y funcionalidad, dentro de los resultados obtenidos a través del uso del motor de búsqueda de la fuente de datos seleccionada.

d) Ejecución de la búsqueda

Como acto seguido, se procedió a definir los criterios para establecer herramientas software candidatas a ser utilizadas en la propuesta *Success Process UCT*.

Para determinar la relevancia de las herramientas de gestión de no conformidades, se crearon algunos criterios de inclusión y exclusión, como se observa en la tabla XXXVII.

TABLA XXXVII. CRITERIOS DE SELECCIÓN DE HERRAMIENTA DE GESTIÓN DE NO CONFORMIDADES.

Criterios de inclusión	Criterios de exclusión
a. Está relacionado con herramientas de gestión de no conformidades. b. Tiene un sitio web. c. Es intuitivo. d. Requiere un esfuerzo menor en capacitación por parte del usuario. e. Se puede mantener sin conocimiento especializado. f. Es software libre	a. Es software propietario. b. Requiere el pago de un servicio. c. Necesita conocimiento especializado para el uso. d. Necesita conocimiento especializado para el mantenimiento

Fuente esta investigación.

Para la aplicación de los criterios de inclusión y exclusión, como parte de la selección, se definieron los filtros que se muestran en la tabla XXXVIII.

TABLA XXXVIII. ESTRATEGIA DE SELECCIÓN.

Filtro	Descripción	Criterio aplicado	
		Inclusión	Exclusión
F1	Buscar el software aplicando la cadena de búsqueda en el motor de la fuente seleccionada.	a	
F2	Identificar la existencia de un sitio web.	b	a
F3	Leer el nombre y año de distribución, aplicando los criterios de inclusión y exclusión.	a	a
F4	Eliminar las herramientas duplicadas.		b
F5	Leer la funcionalidad y aplicar los criterios de inclusión y exclusión.	c, d, e y f	c y d

Fuente: esta investigación.

En la Fig. 20, se pueden observar los resultados de la ejecución de los filtros definidos.

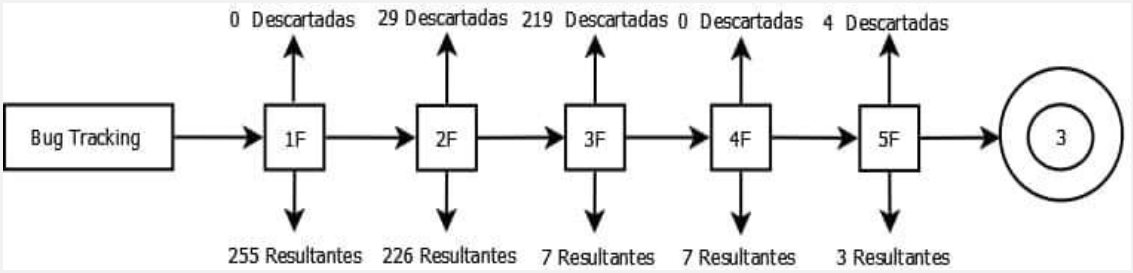


Fig. 20. Mapa de la revisión de herramientas gestión de no conformidades.

Fuente: esta investigación.

Para la gestión de no conformidades, se encontraron las herramientas que se presentan en la tabla XXXIX.

TABLA XXXIX. HERRAMIENTAS DE GESTIÓN DE NO CONFORMIDADES.

Nombre	**Funcionalidad**	**Año última actualización**	**Enlace sitio web**
ZenTao	Herramienta Scrum de código abierto para proyectos de desarrollo de aplicaciones que incluye: *Backlog Management, Sprint And Task, Bug Tracking, Scrum, Waterfall, Roadmap, Burndown, Kanban, SaaS, Self Hosting.*	2020	https://www.zentao.pm/
Mantis BT	Rastreador de errores basado en la web, de fácil implementación que permite el seguimiento de errores del producto software. Requiere PHP, MySQL y un servidor web.	2021	https://www.mantisbt.org/
JTrac	Aplicación web genérica de seguimiento de errores, permite adicionar campos personalizados y menús desplegables. Incorpora flujo de trabajo personalizable, nivel de permisos, integración de correo electrónico, archivos adjuntos y una vista de historial detallada.	2022	http://j-trac.sourceforge.net/

Fuente: esta investigación.

2) Selección de la herramienta de gestión de no conformidades

Posterior a la ejecución de las actividades anteriores, se procedió a seleccionar una herramienta que dé respuesta a la pregunta que guía esta revisión **(RQ1).** Para realizar esta tarea, se procedió a instalar cada una de las herramientas y verificar, a partir del uso el nivel de aporte que realizarían a la gestión de no conformidades (Ver Fig. 20). A partir de los principios generales que propuso [34], y la actualización realizada [35] para definir la medida en la cual una herramienta software puede ser utilizada por usuarios para lograr un objetivo con efectividad, eficiencia y satisfacción en un contexto de uso definido; se procedió a elaborar una matriz de evaluación con el fin de seleccionar una herramienta por cada grupo de búsqueda encontrado. En la TABLA XL, se puede observar la matriz que permite evaluar el uso de cada herramienta.

TABLA XL. MATRIZ DE EVALUACIÓN DEL USO DE LA HERRAMIENTA SOFTWARE.

Criterio	Alto	Medio	Bajo
Relación software-realidad			
Libertad y control			
Reconocer antes que recordar			
Recuperación ante el error			
Ayuda y documentación			

Fuente: esta investigación.

Relación software-realidad: utiliza el lenguaje del usuario, con expresiones y palabras que le resulten familiares. Además, los datos aparecen en un orden lógico y natural.

Libertad y control: cuando elige una opción por error, el usuario debe disponer de una "salida de emergencia" para abandonar el estado no deseado en que se halla.

Reconocer antes que recordar: hace visible acciones y opciones para que el usuario no tenga que recordar información entre distintas secciones.

Recuperación ante el error: presenta mensajes de error claros, permite continuar con la tarea y recuperar lo último realizado.

Ayuda y documentación: la ayuda es fácil de localizar, específica los pasos necesarios y no es extensa.

En la tabla XLI, se puede observar los resultados de la evaluación por cada una de las herramientas encontradas para la gestión de no conformidades, donde **A** es **Alto**, **M** es **Medio** y, **B** es **Bajo**.

TABLA XLI. MATRIZ DE EVALUACIÓN DEL USO DE LA HERRAMIENTA SOFTWARE PARA GESTIÓN DE NO CONFORMIDADES.

Criterio	ZenTao			Mantis BT			JTrac		
	A	**M**	**B**	**A**	**M**	**B**	**A**	**M**	**B**
Relación software-realidad	X			X				X	
Libertad y control		X			X				X
Reconocer antes que recordar			X	X					X
Recuperación ante el error		X			X			X	
Ayuda y documentación			X	X			X		

Fuente: esta investigación.

Para seleccionar la herramienta software, se estableció un esquema de valoración para inspeccionar el nivel de cumplimiento de los criterios de la siguiente manera: Alto (2 puntos), Medio (1 punto) y Bajo (0 puntos). Con base en la puntuación que obtenga cada herramienta software, se procede a calcular el porcentaje a partir del máximo valor posible.

Con la valoración obtenida a partir de la evaluación, se estableció el puntaje y porcentaje para cada herramienta software como se presenta en la tabla XLII.

TABLA XLII. PUNTAJE TOTAL PARA LAS HERRAMIENTAS SOFTWARE DE GESTIÓN DE NO CONFORMIDADES.

Software	**Puntaje**	**Porcentaje**
Mantis BT	8	80%
ZenTao	4	40%
JTrac	4	40%

Fuente: esta investigación.

Finalmente, y teniendo en cuenta los porcentajes obtenidos, la herramienta software seleccionada fue Mantis BT.

3) Diseñar el taller de validación

Para el diseño del taller se realizaron las actividades que se presentan a continuación.

a) Elaboración de artefactos

En primera instancia, se elaboran artefactos para apoyar actividades del proceso *Success Process UCT* que requerían la implementación de los mismos, los cuales se especifican y despliegan con toda la información en el Anexo G.

La técnica que con mayor frecuencia se utiliza para recopilar requerimientos o necesidades de los interesados en el desarrollo ágil de software corresponde con las historias de usuario, según los autores [36][31] [37]. La historia de usuario es una técnica que permite describir una necesidad que será valiosa para un usuario o interesado [38]. Con base en la propuesta hecha por [39], se elaboró un artefacto para especificar la historia de usuario y a través de la conversación con los interesados establecer los criterios de aceptación. En la Fig. 21, se puede observar el formato de especificación.

Historia de Usuario			
Código:	HU-X		
Nombre:			
Actor:			
Descripción:	Como [Actor] quiero [Funcionalidad] para [beneficio]		
Criterios de aceptación:	**CID**	**Condición**	**Resultado**
	1		
	2		
	3		

Fig. 21. Artefacto para especificar historias de usuario.

Fuente: esta investigación.

Según [40] un escenario de prueba asegura el funcionamiento integral del producto software, considerando los flujos comerciales que posee. Para establecer los escenarios de pruebas unitarias, se utiliza el artefacto que se muestra en la Fig. 22, el cual se basa en el/los criterios de aceptación de la historia de usuario; además, se incluyen valores que conforman dicho escenario.

Código HU	
No	
Descripción	
Datos	
CID	

Fig. 22. Artefacto para especificar un escenario de prueba.

Fuente: esta investigación.

Posteriormente se crean los casos de prueba, seleccionando el método a inspeccionar, identificando los datos y escenarios de prueba que corresponden con la validación del mismo. Según [41] un caso de prueba puede ser definido como un elemento para describir los componentes de una prueba. En la Fig. 23 se observa el artefacto de especificación del caso de prueba.

CPId	Nombre	Clase	Método	HU	Escenario	Valores de entrada	Resultado esperado

Fig. 23. Artefacto para especificar un caso de prueba.

Fuente: esta investigación.

A continuación, se presenta un ejemplo de la forma como se especifican los formatos.

En la Universidad de Nariño se requiere llevar un control del esquema de vacunación contra COVID-19 que tiene cada estudiante activo, con el fin de reducir el riesgo de contagio en el retorno a la presencialidad. La HU-001 describe la acción del estudiante frente a la necesidad que presenta la Universidad.

Historia de Usuario			
Código:	HU-001		
Nombre:	Registrar dosis		
Actor:	Estudiante		
Descripción:	Como estudiante quiero registrar una dosis de la vacuna contra el covid-19 que me aplicaron para reducir el riesgo de infección en el retorno a la presencialidad en la Universidad de Nariño.		
Criterios de aceptación:	**CID**	**Condición**	**Resultado**
	1	Cuando se registre la identificación, fecha y tipo de vacuna	Entonces, se debe agregar la dosis del estudiante a la lista
	2	Cuando se digite una identificación de una persona que no es estudiante	Entonces, se debe presentar un mensaje informando que el estudiante no existe
	3	Cuando se digite la identificación de un estudiante que no se encuentra activo en la Universidad	Entonces, se debe presentar un mensaje informando que el estudiante actualmente no se encuentra matriculado

Fig. 24. HU-001 Registrar dosis.

Fuente: esta investigación.

Con base en la historia de usuario descrita en la Fig. 24, se diseña el modelo del mundo, el cual representa las entidades y relaciones que hacen parte del sistema, y servirán de apoyo para dar solución al problema propuesto (Ver Fig.25).

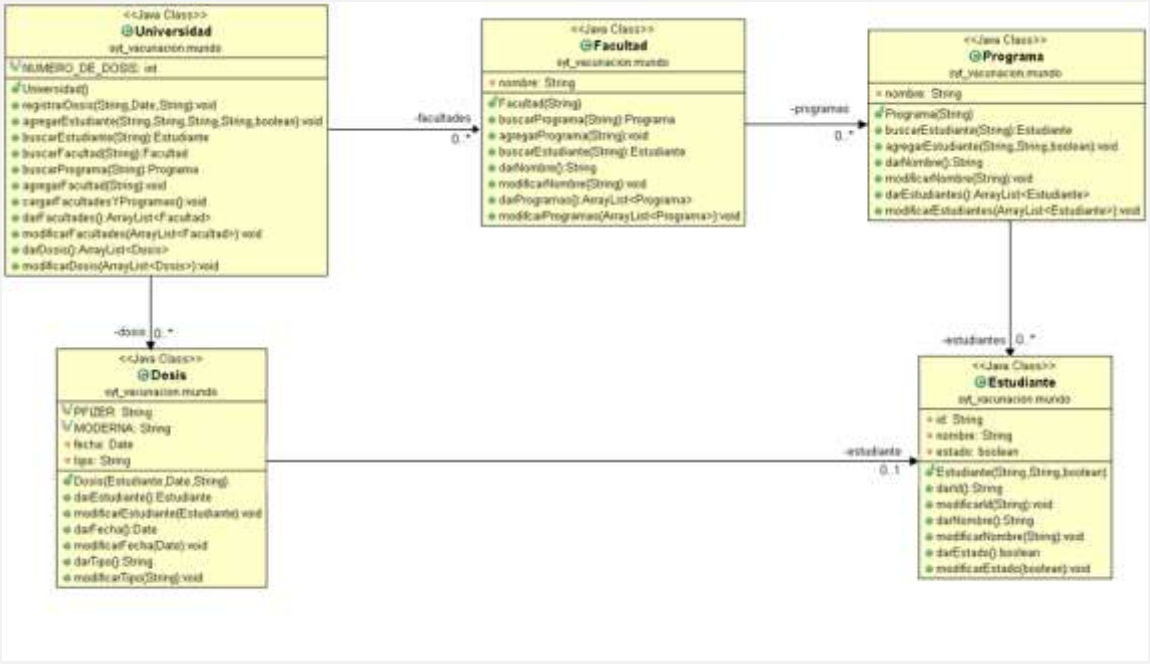


Fig. 25. Diagrama de clases del modelo del mundo.

Fuente: esta investigación.

Se crea el escenario de prueba, en este caso se consideran los criterios de aceptación 2 y 3, de la HU-001, para los cuales la clase Universidad cuenta con el registro de tres estudiantes. La Fig. 26, presenta la información del artefacto diseñado para la especificación del escenario de prueba, encontrando una breve descripción y los valores o datos que lo conforman, en este caso se requiere especificar la facultad a la que pertenece el estudiante, el programa, los atributos código, nombre y estado del estudiante.

Código HU	HU-001
No	2
Descripción	La universidad tiene tres estudiantes
Datos	Estudiante1= {"Facultad de Ingeniería", "Ingeniería de Sistemas", "100"," Adriana", true} Estudiante2= {"Facultad de Ingeniería", "Ingeniería de Sistemas", "101"," Jeferson", true} Estudiante3= {"Facultad de Ingeniería", "Ingeniería de Sistemas", "102"," Samuel", false}
CID	2,3

Fig. 26. Escenario de prueba para HU-001.

Fuente: esta investigación.

Continuando con el proceso planteado en *Success Process UCT*, se construye el/los casos de pruebas unitarias correspondientes al escenario de prueba descrito en la Fig. 22, asignando un nombre descriptivo al caso de prueba. Para ello, se recomienda usar el prefijo *"test"* seguido del nombre del método a validar, registrando los valores de entrada que se utilizarán en la ejecución de la prueba unitaria con su respectivo resultado esperado, como se muestra en la Fig. 27.

CPId	Nombre	Clase	Método	HU	Escenario	Valores de entrada	Resultado esperado
1	testRegistrarDosis1	Universidad	registrarDosis	HU-001	1	Identificación="200" Fecha="7-1-2022" Tipo=" Pfizer"	Falso. El estudiante 200 no existe
2	testRegistrarDosis2	Universidad	registrarDosis	HU-001	2	Identificación="102" Fecha="7-1-2022" Tipo=" Pfizer"	Falso. El estudiante 102 no está activo
3	testRegistrarDosis3	Universidad	registrarDosis	HU-001	2	Identificación="101" Fecha="7-1-2022" Tipo=" Pfizer"	Verdadero. Dosis del estudiante 101 registrada

Fig. 27. Caso de prueba para la HU-001.

Fuente: esta investigación.

b) Instalación y configuración de la herramienta de gestión de no conformidades

A continuación, se describen los pasos para la instalación de la herramienta de no conformidades Mantis *Bug Tracker*.

- El primer paso es descargar la herramienta Mantis *Bug Tracker* de la página *SourceForge* https://sourceforge.net/projects/mantisbt/files/mantis-stable/2.25.2/. Para la ejecución del taller se utilizó la versión 2.25.2

- La herramienta requiere de un servidor web y un gestor de base de datos, por lo tanto, es necesario instalar XAMPP, el cual cuenta con los servicios de Apache y MySQL, en este caso se utiliza la versión 7.4.27, y se puede descargar mediante el siguiente enlace https://www.apachefriends.org/es/download.html

- Finalmente, se debe descomprimir el archivo de descarga de ManitsBT en la ruta C:\xampp\htdocs, e ingresar en la URL del navegador a localhost/NombreCarpetaArchivo para ejecutar su instalación.

Mantis *Bug Tracker* permite la gestión de no conformidades mediante la interacción de usuarios de un proyecto software, a través de la creación de reportes, monitoreo de actividades ejecutadas y envío de notificaciones mediante correo electrónico.

En el Anexo H se encuentran los recursos para la aplicación del taller, referente al proceso de instalación, configuración de Mantis *Bug Tracker* y configuración del ambiente de desarrollo.

4) Ejecutar el taller de validación

En el mes de abril de 2022, se llevó a cabo el taller de validación del proceso de aplicación de pruebas automáticas de unidad y componentes *Success Process UCT*, con los estudiantes de la electiva de Programación Avanzada III del Programa de Ingeniería de Sistemas de la Universidad de Nariño, en el segundo semestre del 2021, en la cual se registran 34 estudiantes matriculados.

El taller tuvo una duración de ocho horas, desarrollado de acuerdo con la siguiente agenda:

1. Presentación del proyecto de investigación y hallazgos significativos del mismo.
2. Explicación de herramientas: automática y gestión de no conformidades.
3. Configuración de ambiente de pruebas y creación de pruebas unitarias.
4. Ejecución de pruebas unitarias.
5. Registro de no conformidad identificada en Mantis *Bug Tracker*.
6. Corrección de no conformidades.
7. Capacitación en el diseño y desarrollo de pruebas automáticas unitarias para el Sprint.

Para la ejecución del taller se facilitó a los estudiantes el proyecto **syt_vacunacion** y una máquina virtual con la instalación y configuración de Mantis *Bug Tracker*. El material construido para el desarrollo del taller y evidencia de la realización del mismo se encuentran disponibles en los siguientes enlaces: https://drive.google.com/drive/folders/14VjrI8QqW9ltd6c04j9vVoX167hcfTsd?usp=sharing

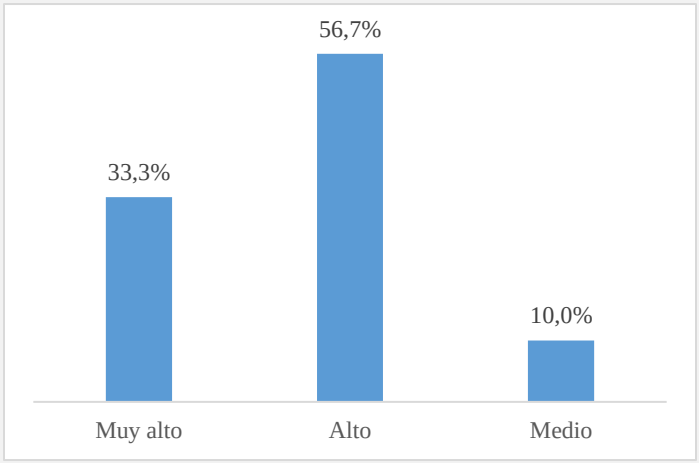
https://drive.google.com/file/d/1pT76-VR2Q7w_-x9LnB2OzDi465dV7oq5/view?usp=sharing

5) Evaluar los resultados de la validación

Para realizar la validación de la propuesta *Success Process UCT* se tuvo como fuente de información a 34 estudiantes de la materia electiva "Programación avanzada III" y se utilizó como técnica de recopilación de datos la encuesta. Posteriormente, se diseñó, elaboró y validó un cuestionario[6]; el cual fue diligenciado por 30 estudiantes. Las variables analizadas corresponden con: nivel de aceptación, aspectos positivos, por incluir y mejorar; de la propuesta.

En la Gráfica 27, se puede observar que la propuesta tuvo un nivel de aceptación del 90,0% en los niveles alto y muy alto, correspondiente a 27 estudiantes, lo que significa que esta forma de abordar las pruebas unitarias automáticas en combinación con el método Scrum ha sido relevante e importante para la formación y ejercicio profesional de los estudiantes.

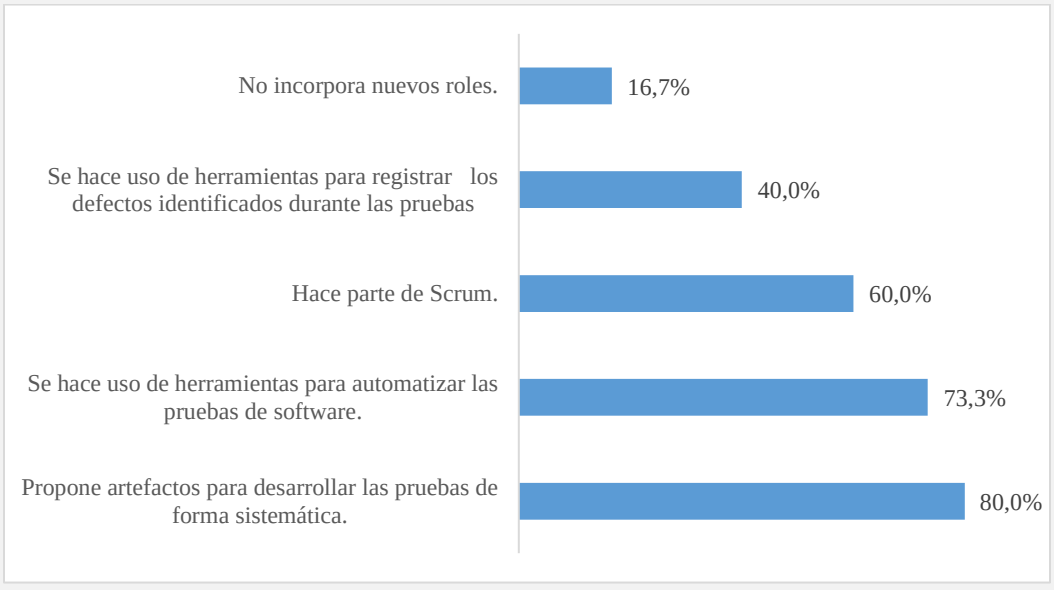
[6] El cuestionario aplicado se puede encontrar en el siguiente enlace:
https://docs.google.com/forms/d/1Jja92edOBN4oy3IjYbqTrpdcU3QUrQ2PdZTXO09DrSI/edit?usp=sharing



Gráfica 27. Nivel de aceptación de la propuesta por los estudiantes.

Fuente: esta investigación.

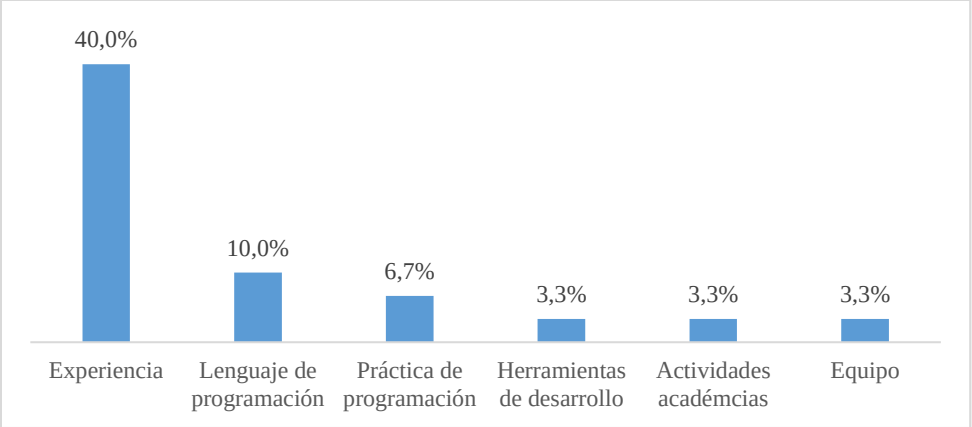
Al indagar por los aspectos positivos de la propuesta, en la Gráfica 28, se puede observar que los estudiantes fundamentalmente resaltan: proponer artefactos para realizar pruebas de forma sistemática, seguido del uso de herramientas para automatizar pruebas, finalizando con incorporar estos aspectos dentro del método Scrum.



Gráfica 28. Aspectos positivos de la propuesta según los estudiantes.

Fuente: esta investigación.

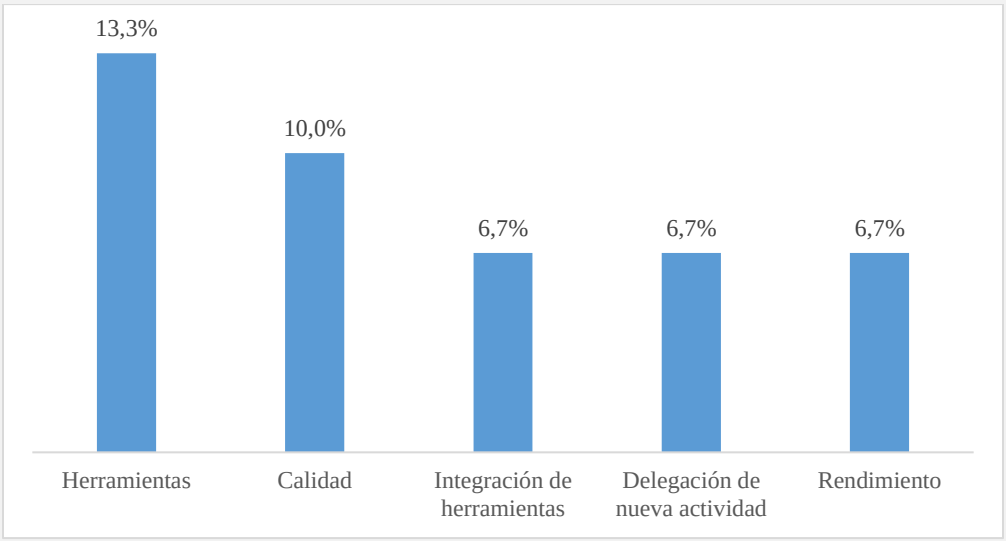
Las dificultades manifestadas por los estudiantes al desarrollar las pruebas unitarias automáticas se presentan en la Gráfica 29, donde se destaca principalmente la experiencia, la cual está relacionada con el razonamiento lógico para diseñar y desarrollar pruebas automáticas y poderlos plasmar en los artefactos para la especificación de la historia de usuario, escenarios y casos de prueba.



Gráfica 29. Dificultades de la propuesta según los estudiantes.

Fuente: esta investigación.

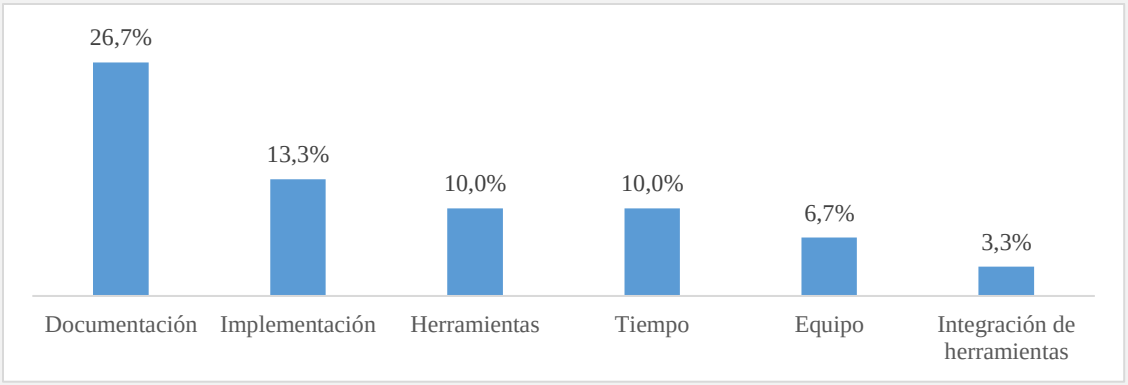
De acuerdo con los datos recolectados, los estudiantes mencionan que uno de los aspectos por incluir en la propuesta, es el uso de herramientas con un 13,3% (Ver Gráfica 30.) correspondiente a cuatro estudiantes. Esta recomendación, se enfoca en incluir software para la gestión del proyecto en Scrum, como por ejemplo Jira o Trello.



Gráfica 30. Aspectos por incluir en la propuesta según los estudiantes.

Fuente: esta investigación.

En cuanto a los aspectos por mejorar, el 26,7% que corresponde a ocho estudiantes, sugieren dar mayor importancia a la documentación que soporta la prueba unitaria; es decir, que exista un contrato claro y preciso en los métodos de las pruebas para que se facilite la comprensión del código elaborado por quién construye las pruebas.



Gráfica 31. Aspectos por mejorar de la propuesta según los estudiantes.

Fuente: esta investigación.

6) Realizar ajustes a la propuesta

Una vez se evalúan los resultados de la propuesta, se efectúan los ajustes de acuerdo con las recomendaciones dadas por los estudiantes, las cuales se describen en la sección: "Evaluar los resultados de la validación". Por esta razón, se elaboran dos procesos como apoyo a la propuesta *Success Process UCT*, que se enfocan en el registro de tareas del *Definition Of Done* (DoD por sus siglas en inglés) y actualización de sus estados.

Para el registro de tareas del DoD, en las tablas: XLIII, XLIV y XLV se presentan los actores, actividades y flujos de control planteados.

TABLA XLIII. ACTORES DEL REGISTRO DE TAREAS DEL DOD.

ID	Nombre del actor	Descripción
AC1	*Scrum Master*	Organiza el *Daily Meeting*, gestiona actividades del equipo y elimina impedimentos.
AC2	*Herramienta para la gestión del DoD*	Permite gestionar las tareas del DoD

Fuente: esta investigación.

TABLA XLIV. ACTIVIDADES DEL REGISTRO DE TAREAS DEL DOD.

ID	Nombre de la actividad	Descripción
ACT1	Establecer las tareas del *Sprint*	Definir las tareas a realizar en el *Sprint*.
ACT2	Estimar las tareas	Calcular tiempo planeado para realizar la tarea.
ACT3	Registrar las tareas	Ingresar tareas a la herramienta.
ACT4	Almacenar la tarea	Guardar la tarea.

Fuente: esta investigación.

TABLA XLV. FLUJO DE CONTROL DEL REGISTRO DE TAREAS DEL DOD.

ID	Descripción	Decisión
CF1	La totalidad de las tareas fueron registradas	Si: FIN No: ACT3

Fuente: esta investigación.

En las tablas: XLVI, XLVII y XLVIII se presentan los actores, actividades y flujos de control planteados en la actualización de estados de las tareas.

TABLA XLVI. ACTORES EN LA ACTUALIZACIÓN DEL ESTADO DE LAS TAREAS DEL DOD.

ID	Nombre del actor	Descripción
AC1	Integrante del equipo Scrum	Miembro del equipo Scrum, encargado de actualizar el estado de la tarea.
AC2	*Herramienta para la gestión del DoD*	Permite gestionar las tareas del *DoD*

Fuente: esta investigación.

TABLA XLVII. ACTIVIDADES PARA LA ACTUALIZACIÓN DEL ESTADO DE LAS TAREAS DEL DOD.

ID	Nombre de la actividad	Descripción
ACT1	Seleccionar tarea del *Sprint*	Identificar tarea del *Sprint*, a la cual se debe cambiar estado.
ACT2	Actualizar estado de la tarea	Modificación de estado actual de la tarea seleccionada.
ACT3	Asignar estado *Doing*	Asignar estado *Doing* a la tarea seleccionada.
ACT4	Asignar estado *Done*	Asignar estado *Done* a la tarea seleccionada.
ACT5	Asignar estado *To Do*	Asignar estado *To Do* a la tarea seleccionada.

Fuente: esta investigación.

TABLA XLVIII. FLUJOS DE CONTROL PARA LA ACTUALIZACIÓN DEL ESTADO DE LAS TAREAS DEL DOD.

ID	Descripción	Decisión
CF1	Tarea en estado *To Do*	Si: ACT3 No: CF2
CF2	Tarea en estado *Doing*	Si: ACT4 No: CF3
CF3	Tarea en estado *Done*	Si: ACT5 No: FIN

Fuente: esta investigación.

Con base en la información descrita en las tablas anteriores, se elaboró el diagrama BPMN de los procesos para la gestión de actividades de Scrum; de acuerdo con las recomendaciones de los estudiantes, mediante la herramienta *Bizagi Modeler*[7], como se observa en las Figuras (Fig.) 28 y 29.

[7] Bizagi Modeler es una marca registrada de la empresa Bizagi que se puede descargar de https://www.bizagi.com/es/plataforma/modeler

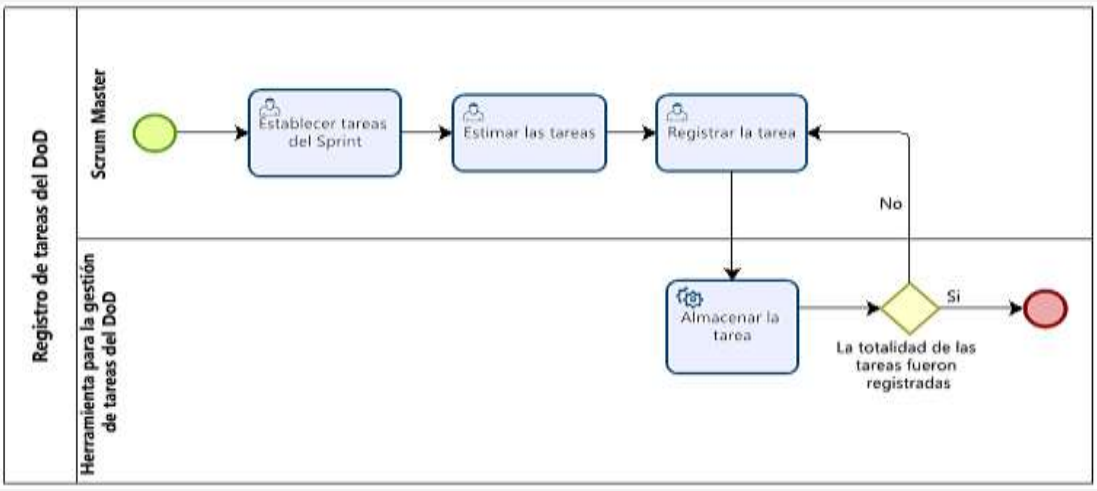


Fig. 28. Diagrama BPMN del proceso para el registro de tareas del DoD.

Fuente: esta investigación.

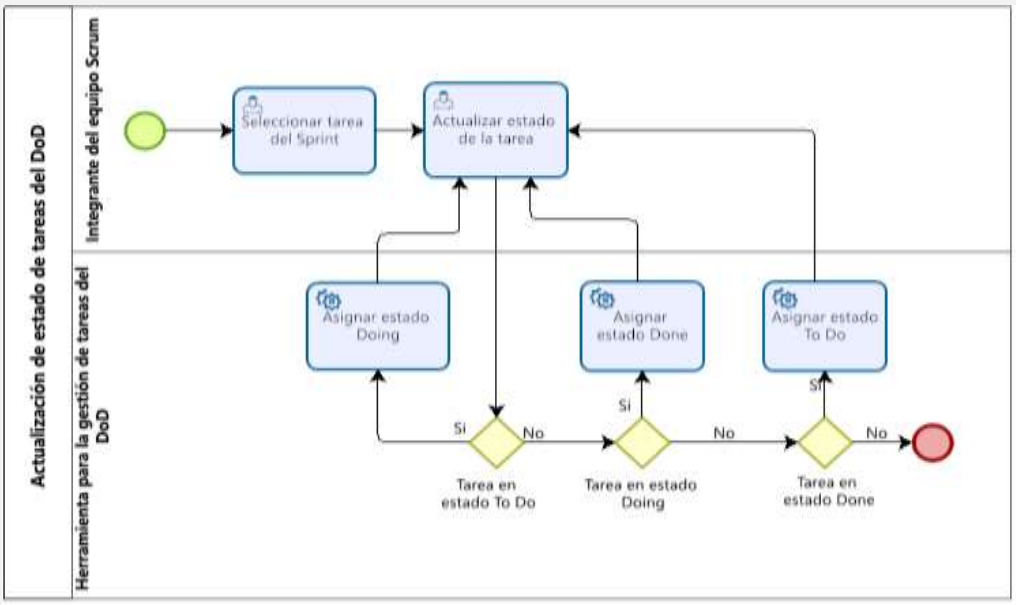


Fig.29. Diagrama BPMN del proceso para la actualización del estado de tareas del *DoD.*

Fuente: esta investigación.

7) Sintetizar resultados

En esta sección se valida la aplicación de la propuesta *Success Process UCT*, seleccionando en primera instancia las herramientas a utilizar. Para la gestión de no conformidades se aplicó una serie de criterios de búsqueda y posteriormente una matriz de evaluación de usabilidad que permitió elegir a Mantis *Bug Tracker*, mientras que la herramienta para la ejecución de pruebas unitarias automáticas se escogió de acuerdo con el lenguaje de programación impartido en la electiva "Programación avanzada III", razón por la cual se utilizó JUnit.

Para realizar el taller de validación, se suministran: artefactos para desarrollar pruebas unitarias, recursos multimedia para la instalación y configuración de la herramienta de gestión de no conformidades, una máquina virtual con Mantis *Bug Tracker* lista para su uso y el caso de estudio a usar en la ejecución del taller.

La validación se realizó con 30 estudiantes, quienes manifiestan la experiencia que tuvieron en relación con la propuesta, obteniendo un nivel de aprobación del 90,0%. Resaltando como aspecto positivo la incorporación de artefactos con un 80,0%. La mayor dificultad para implementar la propuesta *Success Process UCT* fue la experiencia relacionada con el razonamiento lógico para diseñar y desarrollar pruebas automáticas, y plasmarlas en los artefactos para la especificación de la historia de usuario, escenarios y casos de prueba. Los estudiantes sugieren incluir software para la gestión del proyecto en Scrum, y mejorar el contrato en los métodos de las pruebas.

Dadas las recomendaciones plasmadas en las experiencias de los estudiantes, se identifica la necesidad de utilizar una herramienta para la gestión de las actividades en Scrum, por lo cual, se elabora dos procesos para la interacción entre la herramienta y los roles presentes en la propuesta creada.

V. DIVULGACIÓN DE LOS RESULTADOS

Como parte de la apropiación social de conocimiento, esta investigación permitió obtener los siguientes productos:

Una ponencia en el 5to Congreso Andino en Computación, Informática y Educación, realizada entre los días 03 a 05 de noviembre del 2021. En la Fig. 30, se presenta el certificado de participación como ponentes.

Fig.30. Certificación como ponentes.

Fuente: esta investigación.

Para ser aceptada la ponencia, se presentó un artículo (Ver Anexo I), el cual se encuentra en proceso de publicación en el libro de memorias del congreso.

Como productos de formación, se diseña y desarrolla un taller con estudiantes de la Universidad de Nariño del cual se obtienen los productos:

Artefactos

Historia de Usuario			
Código:	HU-X		
Nombre:			
Actor:			
Descripción:	Como [Actor] quiero [Funcionalidad] para [beneficio]		
Criterios de aceptación:	**CID**	**Condición**	**Resultado**
	1		
	2		
	3		

Fig.31. Artefacto para especificar historias de usuario.

Fuente: esta investigación.

Código HU	
No	
Descripción	
Datos	
CID	

Fig.32. Artefacto para especificar un escenario de prueba.

CPId	Nombre	Clase	Método	HU	Escenario	Valores de entrada	Resultado esperado

Fig.33. Artefacto para especificar un caso de prueba.

Fuente: esta investigación.

Videos

- Instalación: https://youtu.be/aC-KGwHIDAM
- Creación de usuarios: https://youtu.be/8mcF2B0eJqI
- Creación de proyectos: https://youtu.be/So8xyNXchaI
- Configuración y envío de correos: https://youtu.be/RUNi8z08THE
- Gestión de no conformidades: https://youtu.be/OSKEUFPjsZo

Caso de estudio

Se realiza el seguimiento al proceso de vacunación de COVID-19 en la Universidad de Nariño, denominado **syt_vacunacion**, el cual se puede obtener del siguiente enlace: https://drive.google.com/file/d/1AqNhZWonVIJc9uumI-6nML_OS051H3UC/view?usp=sharing

Otros recursos

Máquina virtual con la instalación y configuración de la herramienta de no conformidades Mantis *Bug Tracker.* Se puede descargar del siguiente enlace: https://drive.google.com/file/d/1dFGMAs0-uL96LrZ1UPjwke9G1mbk6YiK/view?usp=sharing

VI. CONCLUSIONES

La caracterización realizada en esta investigación, permitió identificar que los proyectos de grado que plantean desarrollar software, presentan en un bajo porcentaje: la planeación, diseño, desarrollo de pruebas automáticas unitarias y la nula aplicación de pruebas de componentes para verificar el funcionamiento y asegurar la calidad de los productos software construidos. Estos resultados se obtienen con base en la evaluación de 289 proyectos de grado, de los cuales 32 proyectos fueron escogidos para su posterior análisis, de acuerdo con criterios y filtros de calidad definidos en la investigación. En este sentido, se abre la posibilidad de proponer un proceso para el desarrollo y ejecución de pruebas automáticas de unidad y componentes, desde el enfoque del desarrollo ágil de software.

Al realizar un mapeo sistemático de literatura, se identificaron 144 artículos relacionados con la ejecución de pruebas automáticas desde el desarrollo ágil de software, encontrando como resultado de la extracción de datos el método AGILUS, las propuestas para IID y Scrum, que se enfocan en el proceso para elaborar e implementar pruebas automáticas ágilmente en proyectos de construcción de software. Lo anterior, evidencia la relevancia de la automatización de pruebas y el creciente interés en el estudio de este campo para incorporarlo en un ambiente profesional.

La propuesta *Success Process UCT* (*Unit And Component Tests*) creada para el desarrollo y ejecución de pruebas automáticas de unidad y componentes, se fundamenta en: la posibilidad de ser aplicada en proyectos de grado con dos o tres integrantes, incorporar artefactos para la planeación y preparación de pruebas y utilizar el método Scrum. Uno de los integrantes del *Development Team* tendrá asignadas actividades para cumplir con funciones de *Testing*, que parten desde la identificación de las historias de usuario, llevar la trazabilidad con el código y elaborar pruebas, apoyándose en herramientas de gestión de no conformidades.

El método se estructura en actividades que abarcan: la planeación, preparación de pruebas, codificación de casos de prueba y aplicación/verificación de los mismos; además, de la gestión de no conformidades y aseguramiento de calidad. La implementación de dicho método, facilita la aplicación de pruebas automáticas de unidad y componentes, haciendo uso de tecnologías presentes en un ambiente laboral, mediante el manifiesto del desarrollo ágil de software, lo que permite incrementar la calidad del producto software y optimizar el tiempo invertido en la fase de validación.

En la validación de la propuesta *Success Process UCT*, se pudo observar que el método tiene un alto nivel de aceptación por parte de los estudiantes, se identifica que la principal dificultad está en el razonamiento lógico para diseñar y desarrollar pruebas automáticas y plasmarlas en los artefactos para la especificación de la historia de usuario, escenarios y casos de prueba. Además, como aspectos por mejorar se encuentran los contratos definidos para los métodos de las pruebas, así como la necesidad de utilizar una herramienta de apoyo para la gestión de las actividades en Scrum. Con base en lo descrito anteriormente, se crean dos procesos que soportan la interacción presente

entre los roles del equipo Scrum y la herramienta, durante el registro y actualización del estado de las tareas del DoD, facilitando a los estudiantes la gestión de dichas actividades, mediante el uso de una tecnología.

VII. RECOMENDACIONES

La investigación permite identificar un incremento en los proyectos de grado que aún no finalizan, por lo que se plantea la necesidad de validar la hipótesis, que dicha situación puede presentarse debido a las nuevas modalidades de graduación, tales como: diplomados, pasantías e incluso el caso de estudiantes que interrumpen el desarrollo de su trabajo de grado, por una oferta laboral. Así mismo, se considera el nivel de complejidad que requiere elaborar un proyecto de grado, y el tiempo empleado para culminarlo, ya que este puede ser afectado debido a factores externos.

En la investigación se plantea una propuesta denominada *Success Process UCT (Unit and Component Tests)* para el desarrollo y ejecución de pruebas automáticas de unidad y componentes, debido a que se encontró un bajo porcentaje de aplicación de las mismas para verificar el funcionamiento y asegurar la calidad de los productos software construidos. En este sentido, se recomienda analizar caminos o formas para que curricularmente se pueda incorporar esta propuesta en la formación de los estudiantes del programa de Ingeniería de Sistemas de la Universidad de Nariño.

Se recomienda indagar los contenidos temáticos de esta investigación para la implementación o preparación de la propuesta *Success Process UCT,* tales como: la metodología de Scrum, los artefactos para la fase de validación, las herramientas de automatización de pruebas, gestión de no conformidades y la gestión de las actividades del DoD en Scrum.

Durante los años 2010 a 2020, se tiene un total de 289 proyectos de grado, de los cuales 72 desarrollaron software, y 32 superaron los criterios de calidad definidos en el protocolo de esta investigación, sin conocer el impacto que han tenido dichos productos software en relación con el mejoramiento de los procesos para los que fueron elaborados. Por esta razón, se recomienda indagar el estado actual del funcionamiento de estos proyectos para con ello poder determinar el impacto que han generado en el medio. Con esta información, se busca identificar, si los criterios de calidad definidos para el despliegue del producto, permitieron su operación, ya sea desde su implementación inicial/actualizaciones, o en su defecto haya sido reemplazado definiendo las condiciones que llevaron a dicha condición.

La propuesta *Success Process UCT* se compone de actividades como: preparación, codificación, verificación de casos de prueba y gestión de no conformidades, apoyadas con la implementación de herramientas que facilitan dichas tareas. Sin embargo, durante la investigación se logra identificar una brecha de interoperabilidad entre las herramientas de desarrollo (Como el *Framework*) y la de gestión de no conformidades. Debido a este hallazgo, se sugiere que se proponga una futura investigación para desarrollar un componente que sirva de interfaz entre estos dos elementos con el fin de integrar las herramientas de codificación y gestión de no conformidades.

VIII. BIBLIOGRAFÍA

[1] R. S. Pressman and University Connecticut, *Ingeniería del software UN ENFOQUE PRÁCTICO*, vol. 7. 2010.

[2] A. Sánchez Upegüi and Fundación Universitaria Católica del Norte., "Introducción: ¿qué es caracterizar?.," 2010.

[3] D. Cohen, M. Linvall, and P. Costa, *Advances in Computers: Advances in Software Engineering*, ELSEIVER A. New York, 2004.

[4] V. Safronau and V. Turlo, "Dealing with Challenges of AutomatingTest Execution," *Proc. Third Int. Conf. Adv. Syst. Test. Valid. Lifecycle*, pp. 14–20, 2011.

[5] J. A. Mera Paz, "ANÁLISIS DEL PROCESO DE PRUEBAS DE CALIDAD DE SOFTWARE," *Ing. Solidar.*, vol. 12, 2016, [Online]. Available: https://doi.org/10.16925/in.v12i20.1482

[6] IEEE Computer Society, *Software Engineering Body of Knowledge (SWEBOK)*. EEUU, 2014.

[7] L. Crispin and J. Gregory, *AGILE TESTING, A PRACTICAL GUIDE FOR TESTERS AND AGILE TEAMS*. 2009.

[8] M. Linares Vásquez, "Charla: La supremacía de los monos, rippers y robots sobre los testers." https://sistemas.uniandes.edu.co/foro/miso/2019/

[9] A. Inc, "https://www.apple.com," 2021. https://www.apple.com/co/iphone/compare/?device1=iphoneSE&device2=iphone12&device3=iphone12mini

[10] G. A. Hernández Pantoja, C. H. Guadir Aza, and A. A. Martinez Navarro, "Web apps characterization, a first step for usability evaluation," *Innov. Dev. Eng. Appl. Sci.*, pp. 1–12, 2019, doi: 10.53358/ideas.v1i1.6.

[11] C. A. Rivera Martínez, "AUTOMATIZACIÓN DE PRUEBAS DE REGRESIÓN," *Univ. Chile*, 2018, [Online]. Available: https://repositorio.uchile.cl/handle/2250/165608

[12] J. Francis, "Los mitos en la automatización de las pruebas.," *Rev. Antioqueña las Ciencias Comput.*, vol. 4, no. 2, pp. 57–60, 2014.

[13] S. Domenico, G. Grano, F. Palomba, F. Filomena, H. C. Gall, and A. Bacchelli, "On the Effectiveness of Manual and Automatic Unit Test Generation: Ten Years Later," *IEEE/ACM 16th Int. Conf. Min. Softw. Repos.*, 2019, doi: 10.1109/MSR.2019.00028.

[14] G. Grano, S. Scalabrino, H. C. Gall, and O. Rocco, "An Empirical Investigation on the

Readability of Manual and Generated Test Cases," *ACM/IEEE 26th Int. Conf. Progr. Compr.*, 2018, doi: 10.1145/3196321.3196363.

[15] Z. Sahaf, V. Garousi, D. Pfahl, R. Irving, and Y. Amannejad, "When to Automate Software Testing? Decision Support Based on System Dynamics: An Industrial Case Study," 2014, doi: 10.1145/2600821.2600832.

[16] M. Gligoric, S. Negara, O. Legunsen, and D. Marinov, "An Empirical Evaluation and Comparison of Manual and Automated Test Selection," 2014, doi: 10.1145/2642937.2643019.

[17] S. N. FERNÁNDEZ VERA, "HERRAMIENTAS DE AUTOMATIZACIÓN PARA PRUEBAS DE SOFTWARE," *Univ. Técnica Federico St. María*, 2018.

[18] L. L. Ochoa, "Automatización de Pruebas de Software: Experiencia y Lecciones Aprendidas," *Univ. Nac. San Agustín*, 2018.

[19] A. Bacchelli, P. Ciancarini, and D. Rossi, "On the effectiveness of manual and automatic unit test generation," *Third Int. Conf. Softw. Eng. Adv.*, 2008, doi: 10.1109/ICSEA.2008.66.

[20] J. Yang, Y. Yuan, and T. Zhang, "Research on Modeling of Software Automatic Test," *Commun. Comput. Inf. Sci.*, 2015.

[21] E. M. Serna, R. M. Martínez, and T. P. Andrea, "Un modelo para determinar la madurez de la automatización de las pruebas del software como área de investigación y desarrollo," 2017.

[22] S. A. White and D. Miers, *GUIA DE REFERENCIA Y MODELADO BPMN*, Future Str. Florida, USA, 2009.

[23] R. Hernández Sampieri, C. Fernández Collado, and M. del P. Baptista Lucio, *METODOLOGÍA DE LA INVESTIGACIÓN*, S.A. DE C. 2014.

[24] R. Landeau, *Elaboración de Trabajos de Investigación*, Editorial. Venezuela, 2007.

[25] Stack Overflow, "Stack Overflow," 2021. insights.stackoverflow.com

[26] Real Academia Española, "Real Academia Española," 2021. https://dle.rae.es/proceso

[27] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic Mapping Studies in Software Engineering," 2008.

[28] R. Cavichi de Freitas, L. A. Rodrigues Jr, and A. M. da Cunha, "AGILUS: A Method for Integrating Usability Evaluations on Agile Software Development," 2016, doi: 10.1007/978-3-319-39510-4_50.

[29] E. Collins, G. Macedo, N. Maia, and A. Dias-Neto, "An Industrial Experience on the Application of Distributed Testing in an Agile Software Development Environment," *IEEE*

Seventh Int. Conf. Glob. Softw. Eng., 2012, doi: 10.1109/ICGSE.2012.40.

[30] S.-T. Lai and F.-Y. Leu, "A Version Control-based Continuous Testing Frame for Improving the IID Process Efficiency and Quality," *10th Int. Conf. Innov. Mob. Internet Serv. Ubiquitous Comput.*, 2016, doi: 10.1109/IMIS.2016.113.

[31] G. Hernández, Á. Martínez, R. Jiménez, and F. Jiménez, *Métricas de productividad para equipo de trabajo de desarrollo ágil de software: una revisión sistemática*, vol. 22. 2019. doi: 10.22430/22565337.1510.

[32] E. A. ARTEAGA CASTILLO and G. A. HERNÁNDEZ PANTOJA, "FACTORES Y ESTRATEGIAS BASADAS EN SOFTWARE LIBRE QUE PROPICIAN EL APRENDIZAJE ORGANIZACIONAL EN EL ÁREA DE TECNOLOGÍA DE LA UNIVERSIDAD DE NARIÑO," vol. 2, no. 2, p. 185, 2018, [Online]. Available: https://doi.org/10.1016/j.gecco.2019.e00539%0Ahttps://doi.org/10.1016/j.foreco.2018.06.029%0Ahttp://www.cpsg.org/sites/cbsg.org/files/documents/Sunda Pangolin National Conservation Strategy and Action Plan %28LoRes%29.pdf%0Ahttps://doi.org/10.1016/j.forec

[33] G. Hernández, Á. Martínez, F. Jiménez, R. Jiménez, and A. Baron, "Learning factory for the Software Engineering area: first didactic transformation," *Proc. - 2021 47th Lat. Am. Comput. Conf. CLEI 2021*, 2021, doi: 10.1109/CLEI53233.2021.9639910.

[34] J. Nielsen, "Enhancing the Explanatory Power of Usability Heuristics," *Proc. SIGCHI Conf. Hum. Factors Comput. Syst. Celebr. Interdepend. - CHI '94*, 1994, doi: 10.1145/191666.191729.

[35] J. Nielsen, "Nielsen Norman Group," 2020. https://www.nngroup.com/articles/ten-usability-heuristics/

[36] G. Hernández, Á. Martínez, I. Argote, and D. Coral, "Metodología adaptativa basada en Scrum: Caso empresas de la Industria de Software en San Juan de Pasto - Colombia," vol. 28, no. Diciembre, pp. 211–223, 2015.

[37] G. Lucassen, F. Dalpiaz, J. M. E. M. van der Werf, and S. Brinkkemper, "Improving agile requirements: the Quality User Story framework and tool," *Requir. Eng.*, vol. 21, no. 3, pp. 383–403, 2016, doi: 10.1007/s00766-016-0250-x.

[38] M. Cohn, *USER STORIES APPLIED FOR AGILE SOFTWARE DEVELOPMENT*. 2004.

[39] K. Villamizar Suaza and Universidad Nacional de Colombia, "Definición de equivalencias entre historias de usuario y especificaciones en UN-LENCEP para el desarrollo ágil de software.," p. 96, 2013.

[40] R. Tommy, N. Prasannakumaran, S. KV, and S. Kesav, "Test Scenario Modeling Modeling Test Scenarios diagrammatically using specification based testing techniques," *Proc. 2015*

Third Int. Conf. Comput. Commun. Control Inf. Technol., 2015, doi: 10.1109/c3it.2015.7060198.

[41] D. Almog and T. Heart, "What Is a Test Case? Revisiting the Software Test Case Concept," *Softw. Process Improv. 13–31*, 2009, doi: 10.1007/978-3-642-04133-4_2.

[42] digital.ai, "15th State of Agile Report," *2021*, 2021.