EFECTOS DE LA METICULOSIDAD COMO DIMENSIÓN DE LA PERSONALIDAD EN LOS ATRIBUTOS DE MANTENIBILIDAD Y FUNCIONALIDAD DEL SOFTWARE EN EQUIPOS DE DESARROLLO

DELGADO JOJOA JUAN DAVID

MAESTRÍA EN INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
DEPARTAMENTO DE SISTEMAS
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE NARIÑO
FEBRERO, 2024

EFECTOS DE LA METICULOSIDAD COMO DIMENSIÓN DE LA PERSONALIDAD EN LOS ATRIBUTOS DE MANTENIBILIDAD Y FUNCIONALIDAD DEL SOFTWARE EN EQUIPOS DE DESARROLLO

Autor
DELGADO JOJOA JUAN DAVID, juan.delgado@udenar.edu.co

Informe final de trabajo de grado presentado como requisito para optar al título de Magíster en Ingeniería de Sistemas y Computación.

Asesor
PhD. OSCAR REVELO SÁNCHEZ

Co-Asesor
Mg. SANDRA VALLEJO CHAMORRO

MAESTRÍA EN INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
DEPARTAMENTO DE SISTEMAS
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE NARIÑO
FEBRERO, 2024

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 asesor

_____
Firma del jurado evaluador

_____
Firma del jurado evaluador

_____
Firma del jurado evaluador

_____
San Juan de Pasto, febrero de 2024

AGRADECIMIENTOS

Agradezco en primer lugar a **Dios**, cuya guía y fortaleza me han acompañado en cada paso de este camino académico.

A mis **padres**, quienes han sido mi pilar fundamental, su apoyo incondicional ha sido la luz que ilumina mi camino.

Expreso mi profundo agradecimiento al Ing **Oscar Revelo**, por su paciencia, dedicación y orientación constante.

También agradezco a la docente **Sandra Vallejo**, por su valiosa contribución y apoyo.

Mi reconocimiento se extiende a todos **mis profesores**, quienes con su conocimiento y enseñanzas han dejado una huella imborrable en mi formación.

DEDICATORIA

A Dios, quien ha sido la fuente inagotable de sabiduría y la guía constante en cada paso de mi vida, le dedico con profundo agradecimiento el fruto de este esfuerzo académico. A mi amada familia, cuyo apoyo incondicional ha sido la columna vertebral de mi trayectoria, les expreso mi más sincero agradecimiento por ser la fuerza que me impulsa a superar nuevos retos en cada etapa. A mi pareja, María Mercedes Morales, le agradezco con todo mi corazón por su amor incondicional, paciencia infinita y constante apoyo durante los momentos de dedicación intensa.

RESUMEN

Estudios han evidenciado que la personalidad influye de forma positiva en las diferentes tareas y/o procesos en el campo de la ingeniería de software. Sin embargo, unos pocos estudios empíricos han evaluado el impacto de los rasgos de personalidad en los atributos de calidad del producto software. El objetivo de este trabajo es evaluar los efectos que tienen la formación de grupos homogéneos bajo la dimensión de la meticulosidad con respecto a la mantenibilidad y la adecuación funcional del producto software. El estudio incluyó un total de 76 participantes de la Universidad de Nariño. De estos, 30 participantes fueron asignados al grupo experimental en la sede de Ipiales, mientras que los restantes 46 participantes conformaron el grupo de control en la sede de Pasto. Para formar los grupos homogéneos, se empleó una herramienta computacional basada en un enfoque de algoritmo genético, considerando los rasgos de personalidad de los estudiantes como criterio de agrupación; los cuales fueron cuantificados a través del cuestionario Big Five Invetory. Para evaluar la mantenibilidad y la adecuación funcional se utilizaron herramientas y enfoques reconocidos en la literatura especializada. Los resultados obtenidos mediante un análisis estadístico indican que no hay una diferencia significativa entre la formación de grupos homogéneos basados en la dimensión de meticulosidad y la formación de grupos según la preferencia estudiantil en cuanto a la mantenibilidad y la adecuación funcional del software. Sin embargo, se observó que los grupos homogéneos obtuvieron valores ligeramente superiores al umbral recomendado y deseado para las variables Coupling Between Objects (CBO) y Response for a Class (RFC), lo que sugiere una posible relación positiva. En relación a la adecuación funcional, la puntuación promedio para Completitud Funcional fue de 0,48 en el Grupo Experimental y 0,50 en el Grupo Control. En cuanto a Corrección Funcional, se obtuvo una puntuación promedio de 0,80 en el Grupo Experimental y 0,81 en el Grupo Control. En términos de Pertinencia Funcional, ambos grupos obtuvieron una puntuación promedio de 0,63. Teniendo en cuenta que 1 es el valor ideal. Estos resultados indican que hay margen para mejorar, aunque no se puede considerar que los valores sean ni altos ni bajos.

Palabras Clave: rasgos de personalidad, meticulosidad, grupos homogéneos, mantenibilidad, adecuación funcional.

ABSTRACT

Studies have shown that personality positively influences different tasks and/or processes in the field of software engineering. However, a few empirical studies have evaluated the impact of personality traits on software product quality attributes. The objective of this work is to evaluate the effects of the formation of homogeneous groups under the dimension of meticulousness concerning the maintainability and functional adequacy of the software product. The study included a total of 76 participants from the University of Nariño. Of these, 30 participants were assigned to the experimental group at the Ipiales headquarters, while the remaining 46 participants made up the control group at the Pasto headquarters. To form homogeneous groups, a computational tool based on a genetic algorithm approach was used, considering the students' personality traits as grouping criteria; which were quantified through the Big Five Inventory questionnaire. To evaluate maintainability and functional adequacy, tools and approaches recognized in the specialized literature were used. The results obtained through a statistical analysis indicate that there is no significant difference between the formation of homogeneous groups based on the dimension of conscientiousness and the formation of groups according to student preference regarding the maintainability and functional adequacy of the software. However, it was observed that homogeneous groups obtained values slightly higher than the recommended and desired threshold for the Coupling Between Objects (CBO) and Response for a Class (RFC) variables, suggesting a possible positive relationship. About functional adequacy, the average score for Functional Completeness was 0.48 in the Experimental Group and 0.50 in the Control Group. Regarding Functional Correction, an average score of 0.80 was obtained in the Experimental Group and 0.81 in the Control Group. In terms of Functional Relevance, both groups obtained an average score of 0.63. Taking into account that 1 is the ideal value. These results indicate that there is room for improvement, although the values cannot be considered to be either high or low.

Key words: personality traits, conscientiousness, homogeneous groups, maintainability, functional adequacy.

TABLA DE CONTENIDO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABLAS

Pág.

ANEXOS

Pág.

GLOSARIO

ADECUACIÓN FUNCIONAL (AF): la capacidad de un sistema o software para cumplir con los requisitos funcionales definidos en su especificación.

BIG FIVE INVENTORY: cuestionario que evalúa los rasgos de personalidad según el modelo de los Cinco Grandes (apertura a nuevas experiencias, meticulosidad, extraversión, amabilidad y neuroticismo).

CASOS DE PRUEBA: conjunto de datos de entrada y resultados esperados para probar la funcionalidad de un programa o sistema.

GRUPOS HOMOGÉNEOS: grupo de individuos que comparten características similares, como habilidades, conocimientos, personalidad, intereses o experiencia.

HU: historias de Usuario, una técnica para definir requisitos de software basada en descripciones narrativas de las interacciones de los usuarios con el sistema.

JAVA: un lenguaje de programación orientado a objetos de alto nivel que se enfoca en la portabilidad y la seguridad en la programación de aplicaciones.

MANTENIBILIDAD: la capacidad de un software para ser modificado y mejorado a lo largo del tiempo, con el fin de corregir errores, mejorar el rendimiento, adaptarse a nuevos requisitos, entre otros.

MÉTRICAS: las métricas son medidas o indicadores cuantitativos utilizados para evaluar o medir un determinado proceso, sistema, producto o servicio.

MÉTRICAS CK: conjunto de métricas que evalúan la calidad del diseño orientado a objetos, incluyendo la complejidad ciclomática, la cohesión, el acoplamiento, entre otras.

METICULOSIDAD: característica de una persona que se preocupa por hacer las cosas de manera cuidadosa, precisa y minuciosa.

PROGRAMACIÓN ORIENTADA A OBJETOS (POO): un paradigma de programación que se enfoca en el uso de objetos y clases para representar conceptos del mundo real, y que permite la encapsulación, herencia y polimorfismo.

SCRUM: un marco de trabajo ágil para el desarrollo de software que se enfoca en la colaboración, la entrega de valor y la adaptabilidad.

INTRODUCCIÓN

La calidad del software es un elemento crucial para el éxito de cualquier proyecto de desarrollo de software. En la actualidad, se ha demostrado que el software de baja calidad puede generar problemas costosos y retrasos en la entrega. Un estudio realizado por CISQ [1], indica que en 2020 el costo de la mala calidad del software en Estados Unidos fue de 2,08 billones de dólares, lo que resalta aún más la importancia de prestar atención a la calidad del software en todas las fases del proceso de desarrollo. La mala calidad del software puede afectar significativamente los costos y la eficiencia de las empresas así como la satisfacción del usuario final [2]. En consecuencia, es fundamental que las empresas inviertan en la selección de profesionales altamente capacitados en el desarrollo de software y en la implementación de prácticas y herramientas de desarrollo que aseguren la calidad del software en todas sus etapas.

En el campo de la Ingeniería de Software (IS), es ampliamente reconocido que el factor humano desempeña un papel crítico en el éxito de los proyectos de desarrollo [3]–[6]. Esto se debe a que no solo los factores técnicos, tecnológicos y metodológicos son determinantes para construir software de calidad [7]. Existen en ese sentido aspectos y características más específicas y de fondo que pueden contribuir a que las tareas de desarrollo de software sean exitosas, como son los rasgos de personalidad, los cuales se refieren al conjunto de características únicas, patrones de pensamiento, sentimientos y comportamientos que definen a una persona [8].

Por lo tanto, numerosos estudios han explorado la influencia de la personalidad en diversos aspectos de la IS, incluyendo la composición de equipos, el rendimiento académico, la productividad del equipo, la asignación de roles, entre otros [9]–[16]. Sin embargo, a pesar de los esfuerzos y avances, aún existen incógnitas, contradicciones y desafíos en esta área [17]–[22]. En particular, se ha prestado una atención limitada a cómo la personalidad afecta atributos específicos de calidad del software [17], [20]–[23]. La complejidad de los seres humanos y las dinámicas en el desarrollo de software plantean desafíos continuos, con incógnitas que persisten en cuanto a la interacción entre múltiples dimensiones de la personalidad y a la manera como estas afectan la IS en diferentes contextos.

El presente estudio aborda este vacío, centrándose específicamente en la dimensión de la meticulosidad y su relación con la mantenibilidad y la adecuación funcional en sistemas orientados a objetos. La meticulosidad, que es una de las cinco dimensiones del modelo Big Five [24], se destaca por su estrecha afinidad con las demandas específicas de la programación de software. Esta dimensión caracteriza a las personas que tienden a prestar mucha atención a los detalles, son ordenadas y organizadas, y se esfuerzan por hacer las cosas de manera precisa y correcta [25], [26], atributos considerados fundamentales para garantizar la calidad del software desarrollado [2], [27]. Además, es importante destacar que la meticulosidad es una de las dimensiones más comunes en los desarrolladores de software [27]. Sin embargo, a pesar de que no se ha explorado extensamente esta dimensión en la literatura, su estudio tiene un gran potencial para aportar al campo, al abordar un aspecto poco explorado y fundamental para comprender el modo en que influye en la calidad del software y en el desempeño de los equipos de desarrollo. Por lo tanto, el objetivo de este estudio es evaluar cómo la formación de grupos homogéneos, basados en la dimensión de la meticulosidad, influye en la mantenibilidad y en la adecuación funcional, en comparación con grupos formados por preferencia de los estudiantes.

Para llevar a cabo lo anteriormente mencionado, se usó una herramienta computacional propuesta por Sánchez et al. [28] para formar grupos homogéneos basados en la dimensión de la

meticulosidad. Dicha herramienta utiliza un enfoque de algoritmo genético que se basa en los rasgos de personalidad de los estudiantes como criterio de agrupación. Como instrumento de medición de los rasgos de la personalidad de los participantes, se aplicó el denominado "Big Five Inventory", un inventario o cuestionario estandarizado basado en el modelo psicológico Big Five. Para evaluar la mantenibilidad y la AF se utilizaron herramientas y enfoques reconocidos en la literatura especializada. Los participantes fueron estudiantes de quinto semestre del programa de Ingeniería de Sistemas de la Universidad de Nariño (sede Pasto e Ipiales), Colombia. Estos alumnos estaban cursando la asignatura de Ingeniería de Software III. El estudio involucró un total de 76 participantes, de los cuales 30 correspondieron a estudiantes de la sede Ipiales (grupo experimental). Por otra parte, 46 estudiantes correspondieron a la sede Pasto (grupo de control).

Este tipo de investigaciones es fundamental por varias razones de gran relevancia. En primer lugar, la meticulosidad es una característica que puede influir en la forma en que los miembros del equipo abordan tareas y resuelven problemas. Al comprender la manera en que esta se relaciona con la composición de equipos, puede facilitar la creación de grupos más afines en términos de enfoque y estilo de trabajo, lo que puede llevar a una mayor cohesión y colaboración. Además, este estudio tiene el potencial de contribuir al desarrollo de técnicas para la formación de equipos de desarrollo de software. Al identificar características de personalidad, como la meticulosidad, que influyen en la dinámica del equipo y en la calidad del trabajo, se pueden establecer criterios más precisos para la selección de miembros del equipo. Esto no solo puede mejorar el rendimiento general de los equipos de desarrollo, sino también la calidad del software producido, lo que a su vez impacta en la satisfacción del cliente y el éxito del proyecto. En este se sentido, se espera fomentar una mayor atención y análisis de estos factores en futuras investigaciones, ya que a menudo los aspectos humanos en la IS son ignorados, no se les presta la misma atención y no se analizan con el mismo detalle que los factores técnicos, tecnológicos y metodológicos [29].

El presente documento organiza el proceso investigativo llevado a cabo de la siguiente forma: el primer capítulo se enfoca en los elementos que dieron origen al proyecto de investigación, como el grupo y la línea de investigación, el planteamiento del problema, la justificación y los objetivos planteados. En el segundo capítulo, se incluye el estado del arte, el marco conceptual que soporta la investigación y la formulación de hipótesis. El tercer capítulo presenta la propuesta metodológica para evaluar los efectos de la formación de grupos homogéneos basados en la dimensión de la meticulosidad frente a la mantenibilidad y AF del producto software. En el cuarto capítulo se exhiben los resultados de la aplicación de la metodología y su correspondiente comparación y análisis estadístico. El quinto capítulo presenta el análisis y discusión de los resultados, mientras que el sexto capítulo se enfoca en las conclusiones finales del estudio. En el séptimo capítulo se presentan las recomendaciones derivadas de una reflexión acerca del trabajo de investigación, mientras que en el octavo capítulo se exponen los trabajos futuros resultantes del estudio. Por último, se listan los referentes de la investigación.

I. CONTEXTUALIZACIÓN

A. GRUPO Y LÍNEA DE INVESTIGACIÓN

El presente proyecto se ha desarrollado en el marco de la línea de investigación de Ingeniería de Software del grupo Galeras.NET, el cual es el primer grupo creado por el Departamento de Sistemas de la Universidad de Nariño en el año 2004. Este grupo se enfoca en la investigación y el desarrollo de adaptaciones tecnológicas y la enseñanza de las ciencias computacionales y la informática aplicada a la educación.

B. PLANTEAMIENTO DEL PROBLEMA

El software es una herramienta cada vez más común en la vida diaria de las personas y las organizaciones, siendo esencial para la eficiencia y competitividad en el mundo empresarial. Además, tiene un impacto significativo en la calidad de vida de las personas en sectores como la salud, la educación, el entretenimiento, entre otros. Por lo tanto, el desarrollo de software de calidad es fundamental para satisfacer las necesidades actuales y futuras de la sociedad.

Para desarrollar software de calidad se requiere habilidades técnicas específicas. Los ingenieros de software deben capacitarse en diversas metodologías, herramientas y técnicas para llevar a cabo sus proyectos de manera efectiva y eficiente [30]. Es importante destacar que estas habilidades son esenciales para el éxito en el desarrollo de software y para garantizar que el producto final cumpla con los estándares de calidad requeridos. Sin embargo, no solo las habilidades técnicas son importantes. Las habilidades blandas, como la personalidad y la capacidad para trabajar en equipo, también son cruciales en el desarrollo de software exitoso [31], [32]. Por ejemplo, la capacidad de análisis, la comunicación efectiva, la gestión de conflictos, el liderazgo y la resolución de problemas son habilidades fundamentales que deben poseer los miembros de un equipo de desarrollo de software para poder trabajar juntos y lograr el éxito en sus proyectos [13], [33].

En los últimos años, se ha prestado cada vez más atención a la personalidad como uno de los componentes clave de las habilidades blandas en el contexto de la IS [14], [22], [28], [34]. La personalidad se refiere al conjunto dinámico y organizado de características que una persona posee y que influyen de manera única en sus cogniciones, motivaciones y comportamientos en diversas situaciones [8], como la meticulosidad, la amabilidad, la apertura a nuevas experiencias, entre otros rasgos.

Diversos estudios se han enfocado en explorar el impacto de la personalidad en aspectos como la toma de decisiones, la satisfacción del equipo, el clima laboral, la calidad del software, el rendimiento académico, la asignación de roles, la productividad del equipo, entre otros. [13]–[16], [25], [27], [33], [35]–[37]. Además, se ha investigado sobre los efectos de personalidades homogéneas y heterogéneas en el rendimiento del equipo. Mientras algunos autores sugieren que los equipos homogéneos funcionan mejor que los equipos con personalidades diversas [14], [25], [28], [38], otros sostienen que los equipos heterogéneos tienen un papel importante en la mejora del rendimiento del equipo [39], [40].

A pesar de la gran cantidad de investigaciones disponibles en la literatura, todavía existen interrogantes sobre qué tipos de personalidad son útiles y beneficiosos para formar un equipo de trabajo ideal [41]. Además, el estudio de la personalidad en el campo de la IS aún se encuentra en una etapa inmadura, con muchas áreas sin explorar y resultados débiles y no concluyentes [20], [41]–[47]. Como resultado, los profesionales pueden encontrarse con muchas ambigüedades al

tratar de aplicar estos estudios en la práctica. Por lo tanto, es un desafío importante para la comunidad científica contribuir, esclarecer y generalizar estos aspectos.

La literatura científica actual presenta una brecha en cuanto a la exploración de la influencia de la personalidad en los atributos de calidad del software, tal y como se ha señalado en estudios previos [21]–[23]. Por lo tanto, el objetivo de este estudio es abordar esta limitación y analizar los efectos de los rasgos de personalidad en la calidad del software. Concretamente, se centrará en examinar el impacto de los grupos homogéneos bajo la dimensión de la meticulosidad con respecto a los atributos de mantenibilidad y la AF del producto software. De tal forma que se pueda contribuir y aportar nuevos conocimientos y clarificando los aspectos importantes para formar grupos de trabajo ideales en la creación de productos de calidad.

La falta de investigación y comprensión de aspectos poco explorados en la IS, como la relación entre la personalidad y los atributos de calidad del software, puede tener consecuencias graves tanto en el ámbito académico como industrial. En la academia, la falta de investigación y conocimiento puede limitar la formación de nuevos profesionales y el desarrollo de nuevas teorías y enfoques en la IS, impidiendo la exploración de nuevas ideas que podrían mejorar el desarrollo de software y la formación de equipos de trabajo ideales. Además, la falta de este tipo de estudios también puede dificultar la transferencia de conocimiento entre la academia y la industria, lo que puede afectar negativamente la innovación y la competitividad en el mercado. Por otro lado, en la industria del software necesita investigaciones más profundas sobre los efectos de la personalidad en los atributos de calidad del software para poder formar equipos de trabajo adecuados y maximizar la calidad del software. Si no existen este tipo de estudios, puede ser difícil determinar qué rasgos de personalidad son importantes para cada atributo de calidad y cómo se pueden utilizar para mejorar la calidad de sus productos. En última instancia, estudios que investiguen la relación entre la personalidad y los atributos de calidad del software son esenciales para garantizar que el factor humano sea considerado y valorado en el desarrollo de software.

En relación con la problemática expuesta, es importante mencionar que la mayoría de las investigaciones realizadas hasta ahora se han llevado a cabo en la cultura occidental, especialmente en los Estados Unidos [20], [41] . Esto se debe a que los estudios realizados en una determinada cultura pueden no ser aplicables a otras culturas debido a las diferencias culturales y sociales. Por lo tanto, se necesitan estudios en países no occidentales para tener una comprensión más completa de la relación entre la personalidad y la IS [20].

Por otra parte, la personalidad es un constructo complejo y multifacético, lo que significa que no es fácil medirlo de manera precisa y objetiva. Además, existen múltiples teorías y modelos de personalidad, cada uno con enfoques y dimensiones diferentes, lo que hace que sea difícil comparar y generalizar los resultados de los estudios [48]. Así mismo, la interpretación de los resultados de las pruebas de personalidad y sus implicaciones prácticas en el trabajo no son sencillas y requieren la intervención de profesionales en psicología o investigadores de áreas afines, que trabajen en conjunto con profesionales de la IS.

También la falta de colaboración entre la comunidad científica y la industria del software, limita la capacidad de aplicar los resultados de la investigación en la práctica. Además, la falta de financiación y recursos para este tipo de investigación también puede limitar su alcance y calidad. Finalmente, es importante destacar que las réplicas de experimentos son necesarias para consolidar un cuerpo de conocimiento consistente que pueda guiar la investigación futura e influir en la

práctica de la IS [20]. Esto permitirá tener una mayor comprensión de la relación entre la personalidad y el campo de la IS, y así poder tomar decisiones informadas y basadas en evidencia.

En este sentido, el planteamiento del problema del presente estudio es: ¿Cuáles son los efectos de la formación de grupos homogéneos bajo la dimensión de la meticulosidad en la mantenibilidad y la AF del producto software?

C. JUSTIFICACIÓN

El estudio propuesto tiene como objetivo investigar los efectos de formar grupos homogéneos bajo la dimensión de la meticulosidad en la mantenibilidad y la AF. Estos factores son críticos en el campo de la IS y clave para el éxito económico de los sistemas [2][49]. Por lo tanto, esta investigación tiene el potencial de contribuir significativamente al desarrollo de sistemas de software más eficientes y efectivos, lo que puede tener un impacto positivo en la industria del software.

La elección de grupos homogéneos en el proyecto propuesto se justifica por varias razones. En primer lugar, se ha demostrado que la formación de grupos homogéneos puede tener un impacto positivo en la eficacia y eficiencia del trabajo en equipo. Esto se debe a que los miembros del equipo homogéneo comparten características similares, como la experiencia, el conocimiento y las habilidades, lo que puede facilitar la comunicación y la toma de decisiones [14], [25], [28], [38].

Así mismo, la dimensión de la meticulosidad se seleccionó debido a su relación con el rendimiento en tareas que exigen atención al detalle, precisión y organización. La construcción de sistemas de software es una tarea compleja, especialmente en lo que respecta a la mantenibilidad y AF. Por tanto, la meticulosidad se considera una factor relevante en la conformación de grupos de desarrollo de software, ya que permite prestar atención a los detalles y asegurar una ejecución precisa de la tarea en cuestión [25], [26], [38], [50], [51].

Por otro lado, la mantenibilidad es un factor crítico en la industria del software, la capacidad de mantener un sistema es una medida de su calidad, y se refiere a la facilidad con la que se pueden realizar modificaciones al sistema sin afectar su funcionamiento [52]. Se estima que el 70% del tiempo de mantenimiento se dedica a comprender el código [53]. Además, la mantenibilidad es reconocido como un criterio de calidad muy importante en el éxito económico de los sistemas y productos de ingeniería [49]. Con respecto a la AF, también es considerado como un factor vital en la industria del software [54], ya que se refiere a la capacidad del software para cumplir con las necesidades del usuario final. Un software que no cumple con las necesidades del usuario final puede ser inútil y no cumplir con su propósito original. Las empresas dedicadas al desarrollo de software, a menudo presentan problemas en el que las características del producto desarrollado no se ajustan completamente a las especificaciones del cliente. En otras palabras, las funcionalidades del producto no cumplen con los requisitos establecidos [55]–[57].

En este sentido, el presente estudio puede aportar tanto a nivel teórico como metodológico. Desde un punto de vista teórico, puede ayudar a comprender mejor cómo influye la meticulosidad en los grupos de desarrollo tanto en la mantenibilidad como en la AF. Desde un punto de vista metodológico, el estudio puede contribuir al desarrollo de técnicas más eficaces para la formación de grupos de desarrollo de software, lo que puede mejorar la calidad del software producido y reducir los costos asociados al mantenimiento.

Así mismo, este estudio tiene implicaciones sociales y económicas. En la actualidad, la mayoría de las organizaciones dependen del software para llevar a cabo sus operaciones diarias. Un software defectuoso puede causar daños financieros y reputaciones significativos, lo que puede afectar negativamente a las organizaciones y, en última instancia, a la sociedad en su conjunto. Si el estudio logra producir resultados significativos, podría ayudar a reducir los costos asociados al mantenimiento del software y mejorar la calidad del software producido, lo que tendría un impacto positivo tanto en la economía como en la sociedad en general.

Finalmente, es importante destacar que la colaboración y el apoyo de la Universidad de Nariño (Colombia), son fundamentales para el éxito del presente estudio. Gracias a su colaboración, se podrán llevar a cabo todas las etapas de la investigación de manera efectiva y eficiente, teniendo acceso a sus instalaciones y recursos, lo que permitirá una recolección de datos más precisa y confiable. Asimismo, se contará con la participación activa de expertos en el área de la IS que aportarán su conocimiento y experiencia en el diseño y desarrollo del estudio, asegurando la rigurosidad metodológica y la validez de los resultados obtenidos.

D. OBJETIVOS

1) Objetivo general

Identificar la incidencia de la dimensión de meticulosidad en la mantenibilidad y a adecuación funcional del producto software en equipos de desarrollo.

2) Objetivos específicos

- Caracterizar los modelos psicológicos empleados para identificar los rasgos de personalidad de los desarrolladores de software.

- Caracterizar los modelos y las métricas de calidad del producto software para evaluar la adecuación funcional y la mantenibilidad del producto software.

- Evaluar los efectos de la meticulosidad en la adecuación funcional y la mantenibilidad del software desarrollado por grupos de trabajo, a través de una actividad académica en cursos de ingeniería de software.

II. MARCO TEÓRICO

A. ESTADO DEL ARTE

En esta sección, se realiza una revisión de estudios que han explorado la influencia de los rasgos de personalidad de los individuos en el campo de la IS. Estos estudios han aportado valiosos hallazgos y experiencias que permiten establecer un marco de referencia para analizar y validar el estudio que se llevará a cabo. En particular, se enfocará en los efectos de la personalidad en los atributos de calidad del software. La revisión del estado del arte permitirá identificar las principales tendencias, limitaciones y oportunidades en este campo de investigación.

El estudio de S. Romano et al. [16] se enfocó en la relación entre los rasgos de personalidad (es decir, apertura, meticulosidad, extraversión, amabilidad y neuroticismo) de 30 estudiantes de licenciatura en Ciencias de la Computación (CS) con respecto a su productividad, así como la calidad interna de los programas que desarrollaron individualmente en una tarea de implementación. Los resultados del estudio indican que las dimensiones de la personalidad como meticulosidad, extraversión y neuroticismo, están significativamente correlacionados con las métricas McCabe y Halstead. Sin embargo, no pudieron demostrar que los rasgos de personalidad están significativamente correlacionados con la productividad de los desarrolladores al implementar programas en Java.

En un estudio llevado a cabo por S. Barroso et al. [58], se investigó la posible relación entre la personalidad de los desarrolladores y la calidad del software que producen. Los investigadores realizaron un experimento controlado con 15 desarrolladores de software que trabajan en una institución educativa, recolectaron datos del test psicológico a través del modelo Big Five y las métricas del software fueron evaluadas por medio de las métricas CK. Los resultados mostraron que solo la métrica DIT (Depth of Inheritance Tree) fue influenciada por los factores de extraversión y neuroticismo. En otro estudio del mismo autor [23], se analizó el impacto de los tipos de personalidad de los estudiantes en la complejidad del software, utilizando las métricas CK y el modelo MBTI para evaluar el perfil de personalidad. Los resultados mostraron que la personalidad tuvo un impacto significativo en la métrica DIT. Además, de los 16 tipos de MBTI posibles, la muestra incluyó 12 tipos, de los cuales 7 eran extrovertidos y 5 introvertidos. Es importante destacar que, en ambos estudios, el experimento fue aplicado de forma individual a los estudiantes. Finalmente coinciden en la necesidad de realizar más investigaciones para profundizar en la relación entre la personalidad y la calidad del software. Se sugiere aumentar el número de participantes y cambiar el entorno de desarrollo, por ejemplo, la ubicación geográfica.

Cabe considerar por otra parte el trabajo de S. Acuña et al. [59], donde analizaron las relaciones entre tres características de desarrollo (factores de personalidad, características de la tarea y procesos del equipo) y dos características de salida (calidad del software y satisfacción de los miembros del equipo). El estudio involucró 35 equipos de estudiantes. Sus resultados revelaron que los equipos más satisfechos con su trabajo son precisamente aquellos que obtienen puntajes más altos en los factores de personalidad de amabilidad y meticulosidad. En consecuencia, la extraversión está significativamente relacionada con una mejor calidad del software para el desarrollo de software siguiendo una metodología ágil. El producto de software fue evaluado mediante una fórmula general que incluyó la descomposición y modularización, capacidad de prueba, funcionalidad, reutilización, estilo de programación y participación de los miembros del equipo.

Weilemann y Brune [21], llevaron a cabo una revisión de literatura sistemática con el objetivo de comprender qué distintos factores humanos o rasgos de personalidad influyen en los diferentes atributos de calidad del software. En dicha revisión, examinaron 106 artículos relevantes y concluyeron que la mayoría de los autores investigaron la influencia de la personalidad en el rendimiento, satisfacción y productividad del equipo, así como en la calidad del software en general. Los resultados resaltaron la necesidad de realizar más estudios en esta área, incluyendo grupos más grandes y sujetos de estudio expertos. Además, aludieron que los estudios futuros podrían centrarse en un aspecto particular de la calidad del producto de software y definir claramente las métricas que se utilizaron para medir ese aspecto distintivo de la calidad del producto software. Así mismo, se podrían enfocar en roles particulares de las personas en un proceso de IS, por ejemplo, arquitectos, programadores, probadores, entre otros.

Hasta donde se sabe por la literatura, no se han llevado a cabo más estudios que evalúen los efectos de la personalidad en los distintos atributos de calidad del software. No obstante, se identificaron otros estudios relacionados que presentan puntos de intersección con el presente trabajo, los cuales se describen a continuación.

En el trabajo de Salleh et al. [60], estudiaron los efectos de la meticulosidad en la eficacia de la programación en pareja como herramienta pedagógica. Utilizaron estudiantes universitarios de pregrado que asistieron a un curso de introducción a la programación. Como hallazgos se encontró que la meticulosidad no afectó significativamente el rendimiento académico, pero la apertura a nuevas experiencias sí presentó una correlación significativa con el rendimiento de los estudiantes emparejados. Así mismo, Gulati et al. [37] llevaron a cabo un estudio en el que participaron 66 estudiantes para evaluar el efecto del temperamento y la personalidad en el rendimiento de los programadores. Utilizando la herramienta clasificadora de temperamento Keirsey (KTS), encontraron una fuerte relación entre el desempeño de los programadores y el temperamento “Guardianes”; quienes se caracterizan por ser personas concretas y cooperadoras, preocupadas por la responsabilidad y el deber, y que buscan la seguridad y el sentido de pertenencia. Sin embargo, no encontraron relación alguna entre desempeño de los programadores y la personalidad, la cual fue evaluada por la prueba de personalidad de Goldberg's IPIP la cual evalúa las cinco dimensiones del modelo Big Five. Por su lado, Hannay et al. [61] evaluaron el efecto de los rasgos de personalidad en las habilidades de programación de los programadores en pareja. La personalidad fue evaluada a través de modelo Big Five. Los resultados sugieren que, en general, los rasgos de personalidad tienen un valor predictivo modesto en el rendimiento de la programación en pareja en comparación con otros factores como la experiencia, la complejidad de las tareas y el país. Los autores concluyen que se debe prestar más atención a investigar otros predictores de rendimiento y los efectos de la personalidad en la colaboración para mejorar el rendimiento en programación en parejas. El trabajo de Salleh et al. [62], realizó una investigación para analizar los efectos de la personalidad en la programación en pareja (PP) como herramienta pedagógica. Se realizaron cinco experimentos centrados en los rasgos de personalidad de meticulosidad, neuroticismo y apertura a nuevas experiencias. Los resultados mostraron que la apertura a nuevas experiencias fue importante para el rendimiento académico de los estudiantes emparejados, mientras que la meticulosidad y el neuroticismo no presentaron un efecto significativo. Además, la PP aumentó la satisfacción, confianza, diversión y motivación de los estudiantes.

En el estudio de Pieterse et al. [39], se utilizó KTS para examinar el impacto de la personalidad en el desempeño del equipo. Sus resultados indicaron que los equipos heterogéneos tuvieron un mejor desempeño que los equipos homogéneos, especialmente durante las primeras fases de crecimiento

del equipo. En el estudio de Sfetsos et al. [63], se evaluó la programación en pareja desde la perspectiva de las personalidades y temperamentos de los desarrolladores y cómo afecta su eficacia. Se realizó un experimento con 70 estudiantes de pregrado para comparar pares con personalidades y temperamentos heterogéneos con pares homogéneos en términos de efectividad de pares. Los resultados indican que las parejas con personalidades y temperamentos heterogéneos tuvieron una mejor comunicación, desempeño y viabilidad de colaboración. Así mismo, Choi et al. [64] realizaron un estudio empírico en el que participaron 68 estudiantes de pregrado y 68 estudiantes de posgrado de maestría para investigar el impacto de la homogeneidad y heterogeneidad del equipo en la productividad de la programación en parejas utilizando el MBTI para evaluar los perfiles de personalidad. Los resultados indicaron que las parejas con personalidades heterogéneas fueron más efectivas que las parejas con personalidades similares.

Por otra parte, Sánchez et al. [28] propone una técnica basada en algoritmos genéticos para formar grupos homogéneos de estudiantes en escenarios de aprendizaje colaborativo, considerando los rasgos de personalidad como criterio de agrupación. Los resultados mostraron que los grupos homogéneos generados por la técnica propuesta produjeron mejores resultados académicos. La investigación destaca la importancia de la formación de grupos homogéneos en escenarios de aprendizaje colaborativo y sugiere que los rasgos de personalidad pueden ser un criterio efectivo para la formación de grupos en este contexto. De igual manera, Karn et al. [65] evaluaron el impacto de los tipos de personalidad MBTI en la efectividad del equipo en proyectos XP y encontraron que el perfil de personalidad de los equipos de IS es muy importante. Además, hallaron que las personalidades tienen un impacto significativo en el desempeño de los equipos, destacando que los equipos con personalidades homogéneas eran más competitivos. Por otro lado, Peeters et al. [38] llevaron a cabo un estudio utilizando el modelo de los Cinco Grandes Factores (FFM) para examinar el impacto de la composición del equipo en el desempeño del equipo. Sus resultados indicaron que los equipos homogéneos tuvieron un mejor rendimiento para resolver tareas estructuradas, mientras que los equipos heterogéneos fueron más útiles para resolver tareas no estructuradas.

Yilmaz et al. [33] llevó a cabo una encuesta a profesionales. Utilizaron FFM como una herramienta para la evaluación de la personalidad. Sus resultados indicaron que un equipo con amabilidad, estabilidad emocional, mayor extroversión y meticulosidad se desempeñó mejor que otros equipos. En el trabajo de Kichuk y Wiesner [66], realizaron un estudio para evaluar las relaciones entre los factores de personalidad de FFM y el desempeño del equipo para los equipos de diseño de productos. Se encontró que los equipos con mayor amabilidad, mayor extraversión, mayores niveles de capacidad cognitiva y menor neuroticismo tenían más éxito. Feldt et al. [66] realizó un estudio para analizar la relación entre los puntos de vista y el comportamiento de 47 desarrolladores profesionales de software. Usaron FFM para evaluar la personalidad de los practicantes. Sus resultados mostraron que los ingenieros de software que tienen atributos de personalidad similares tienen los mismos puntos de vista. En el trabajo de Kanij et al. [66], utilizaron FFM para evaluar el efecto de los atributos de personalidad en el desempeño de los probadores de software. Se encontró que el desempeño de los evaluadores tenía una correlación positiva con los atributos de personalidad de extraversión y meticulosidad. Así mimo, Shameem et al. [62] realizaron un estudio exploratorio que buscaba desarrollar las mejores combinaciones posibles entre las diferentes dimensiones de la personalidad de los miembros del equipo y los factores críticos del clima del equipo en el contexto del desarrollo de software. Se utilizó FFM para medir la personalidad individual y el Inventario de Clima de Equipo (TCI) para medir el clima del equipo. El estudio

propone que la meticulosidad es importante para el resultado visionario y la orientación a la tarea, mientras que la personalidad extrovertida es importante para mejorar la seguridad participativa y la innovación en la tarea.

Al revisar el estado del arte en el estudio de la personalidad y su relación con los atributos de calidad del software, así como los estudios relacionados con el presente trabajo, se han identificado varias áreas de oportunidad de investigación. Los estudios existentes sobre los efectos de la personalidad en los atributos de calidad del software son limitados, se necesitan más estudios para ampliar y/o generalizar los hallazgos. Además, estos estudios no han explorado cómo diferentes roles dentro del proceso de IS pueden influir en los atributos del software. Asimismo, las muestras utilizadas en los estudios han sido pequeñas y se han aplicado de forma individual, limitando su generalización. Para mejorar la investigación, es necesario aumentar el número de participantes y variar el entorno de desarrollo, por ejemplo, la ubicación geográfica. Es importante destacar que el desarrollo de software se realiza en equipo, ya que fomenta la colaboración y el intercambio de conocimientos entre los miembros del equipo, lo que puede mejorar la calidad del producto final. Además, es valioso destacar que cada dimensión de personalidad puede tener un impacto diferente en los atributos de calidad del software. Por lo tanto, evaluar cada dimensión por separado permite obtener una comprensión más precisa y detallada de la relación entre la personalidad y los atributos de calidad. Este enfoque puede ser especialmente útil para la selección y evaluación de los desarrolladores de software en el futuro, ya que permitiría identificar las dimensiones de personalidad que se correlacionan positivamente con la calidad del software y, por lo tanto, podrían ser deseables en los candidatos a desarrolladores de software. Finalmente, es importante destacar que el estudio de la relación entre la personalidad y los atributos de calidad del software es todavía un campo en desarrollo y aún no se ha llegado a una conclusión definitiva sobre las tendencias en esta relación. De acuerdo con las referencias [20]–[23], es necesario seguir investigando para poder comprender mejor cómo la personalidad puede afectar la calidad del software.

B. SUPUESTOS TEÓRICOS

En el siguiente capítulo se presentarán los supuestos teóricos que abordan los temas relacionados con la personalidad y los grupos, así como conceptos básicos sobre calidad del software, y los atributos de mantenibilidad y adecuación funcional, los cuales son fundamentales para la comprensión de la investigación realizada. En la Fig. 1 se presenta un mapa mental de este marco conceptual.

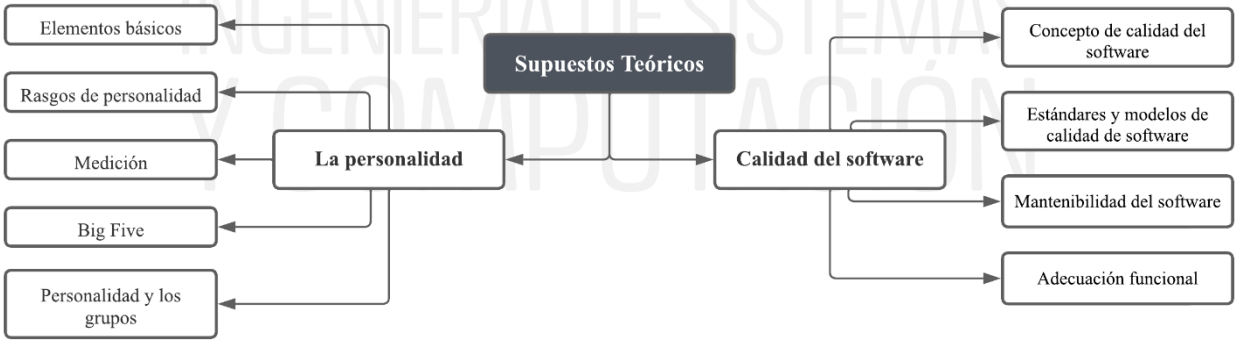


Fig. 1. Mapa mental de los supuestos teóricos.

Fuente: esta investigación.

1) La personalidad

En esta sección se presentan algunos elementos fundamentales sobre la personalidad desde la perspectiva psicológica. Se explorará cómo la personalidad influye en la formación de grupos, y cómo esto, a su vez, afecta su desempeño. También se describirá cómo se mide la personalidad. Por último, se presentará una aproximación al Modelo Big Five, el cual es el modelo utilizado en este trabajo para cuantificar los rasgos de personalidad.

2) Elementos básicos sobre personalidad

La personalidad es un concepto importante y estudiado en diversas corrientes psicológicas. A lo largo de la historia, se ha buscado describir y entender la personalidad, pero la falta de criterios universales para identificar perfiles de personalidad genera controversia en su definición. A pesar de esto, su relevancia teórica y las manifestaciones evidentes en los seres humanos demuestran que sigue siendo un tema relevante en la psicología [64].

La gran cantidad de enfoques que existen sobre la personalidad hacen que este concepto sea de gran importancia para diversas corrientes psicológicas. A lo largo de la historia, la teoría de la personalidad ha sido objeto de estudio y ha sido mejor descrita. Sin embargo, la falta de criterios universales para identificar perfiles de personalidad hace que sea un constructo controvertido. A pesar de esto, la importancia teórica que ha tenido en la investigación a lo largo de la historia y las manifestaciones evidentes en los seres humanos demuestran que sigue siendo un tema relevante en la psicología [67].

Existen múltiples definiciones del término personalidad propuestas por varios psicólogos, las cuales incluyen elementos esenciales para la conceptualización teórica del constructo [68]. A pesar de ello, no existe una definición perfecta de personalidad, ni tampoco existe un consenso generalizado en el campo de la psicología. Aunque un debate más profundo sobre la terminología y las definiciones conceptuales excede el alcance de este documento, es necesario contar con algunos elementos básicos para orientar el proceso. En esta sección, se proporcionan dichos elementos.

En general, se considera que la personalidad es una organización dinámica de sistemas psicofísicos que crean los patrones característicos de comportamiento, pensamiento y sentimiento de una persona. Según Ryckman [8], la personalidad es el conjunto dinámico y organizado de características que posee una persona, que influyen de forma única en sus cogniciones, motivaciones y comportamientos en diversas situaciones. Esta definición separa claramente la personalidad de otros constructos como la cognición, la motivación y el comportamiento, que no son de interés central en este trabajo.

Delgado et al. [69] presentan tres perspectivas diferentes de la personalidad: a) como una organización total de las tendencias reactivas, patrones de hábitos y cualidades físicas que determinan la efectividad social del individuo; b) como un modo habitual de ajustes que el organismo efectúa entre sus impulsos internos y las demandas del ambiente; y c) como un sistema integrado de actitudes y tendencias de conductas habituales en el individuo que se ajustan a las características del ambiente.

Por su parte, Allport [70] asume que la personalidad se refiere a la integración de todos los rasgos y características del individuo que determinan una forma de comportarse. Es decir, la personalidad se forma en función del desarrollo del individuo, a partir de las características ambientales, biológicas y sociales que explican, modulan y mantienen su comportamiento.

La personalidad se compone de varios términos importantes, como el temperamento y el carácter. Según Allport [70], el temperamento se refiere a las emociones naturales que pueden ser influenciadas por factores genéticos y hereditarios. Por otro lado, el carácter se relaciona con el grado de organización moral de un individuo, que se basa en juicios de valor y evaluaciones éticas. El carácter es moldeado por la experiencia de vida de cada persona y controla, modifica, corrige y autorregula la actividad de los individuos para responder a las exigencias del medio, como afirma Josep [71]. Cada individuo adquiere una combinación única de sentimientos y valores a lo largo de su desarrollo, lo que hace que el carácter difiera en cada persona en función de su perspectiva y forma de interpretar la realidad humana.

3) Rasgos de la personalidad

Según Allport [70], el rasgo es la unidad primaria de la personalidad. Un rasgo se define como una característica relativamente estable de la personalidad que influye en el comportamiento de las personas. La teoría de los rasgos de la personalidad es una de las principales áreas teóricas en el estudio de la personalidad, la cual sugiere que la personalidad de un individuo está compuesta por una amplia variedad de factores que influyen en su comportamiento.

A diferencia de otras teorías de la personalidad, como las teorías psicoanalíticas o humanísticas, la teoría de los rasgos de la personalidad se enfoca en las diferencias entre los individuos. La personalidad única de cada individuo se forma a partir de la combinación e interacción de diversos rasgos. Por lo tanto, la teoría de los rasgos se centra en la identificación y medición de estas características individuales de la personalidad.

A continuación, se presentan de forma resumida algunos de los autores más influyentes en la fundamentación de la teoría de los rasgos [72].

Teoría de los rasgos de Gordon Allport

En 1936, el psicólogo Gordon Allport descubrió que un solo diccionario de inglés contenía más de 4.000 palabras que describen diferentes rasgos de personalidad, los cuales categorizó en tres niveles:

- **Rasgos cardinales**: Son los que predominan en toda la vida de un individuo, llegando a ser tan representativos que se les asocian directamente a estas características. Son los rasgos que más influyen y dan forma al comportamiento de la persona. Algunos ejemplos específicos son el narcisismo o el Don Juan.
- **Rasgos centrales**: Son las características generales que forman los fundamentos básicos de la personalidad. Aunque no son tan dominantes como los rasgos cardinales, sus características principales pueden aplicarse a numerosas personas. Términos como inteligente, honesto, tímido o ansioso son considerados rasgos centrales.
- **Rasgos secundarios**: Son los rasgos que a veces están relacionados con actitudes o preferencias y suelen aparecer solo en ciertas situaciones o bajo circunstancias específicas. Algunos ejemplos son “se pone muy nervioso cuando habla delante de varias personas”, “es impaciente cuando tiene que esperar” o “le gusta esto o aquello”.

Los dieciséis tipos de personalidad de Raymond Cattell

La teoría de los rasgos de la personalidad de Raymond Cattell [73] simplificó la lista inicial de más de 4.000 rasgos de Allport a solo 171, eliminando aquellos poco comunes y combinando características similares. Clasificó a una amplia muestra de individuos en estos 171 rasgos diferentes y, utilizando el análisis factorial, identificó términos estrechamente relacionados, reduciendo su lista a solo 16 rasgos de personalidad. Cattell afirmó que estas 16 características son la base de toda la personalidad humana.

Las tres dimensiones de la personalidad de Eysenck

El psicólogo británico Hans Eysenck [74] propuso un modelo de personalidad basado en tres dimensiones o factores principales: introversión/extraversión, neuroticismo/estabilidad emocional y psicoticismo. La dimensión de introversión/extraversión se refiere a la dirección de la atención hacia las experiencias internas o hacia el medio ambiente y otras personas. Por ejemplo, una persona que puntúa alto en introversión podría ser tranquila y reservada, mientras que una persona que puntúa más en extraversión podría ser sociable y expansiva. La dimensión de neuroticismo/estabilidad emocional se relaciona con la tendencia de un individuo a experimentar inestabilidad emocional o a mantenerse emocionalmente constante y estable. Finalmente, la dimensión de psicoticismo se refiere a la capacidad de una persona para hacer frente a la realidad y se caracteriza por tendencias antisociales, hostiles, falta de empatía y manipulación en las personas que tienen un alto contenido de este rasgo.

La teoría de los cinco factores de personalidad

La teoría de Eysenck y la de Cattell han sido ampliamente investigadas, lo que ha llevado a algunos teóricos a sugerir que Cattell se centró en una cantidad excesiva de rasgos, mientras que Eysenck se concentró en muy pocos. Como resultado, ha surgido una nueva teoría de los rasgos, comúnmente conocida como la teoría de los “Cinco Grandes” (Big Five). Este modelo se basa en cinco rasgos fundamentales que interactúan para formar la personalidad humana, más adelante se dará detalle de este modelo.

4) Medición de la personalidad

A medida que se ha buscado comprender el concepto de personalidad para explicar el comportamiento humano, se ha generado la necesidad de crear herramientas de medición que permitan evaluar las características individuales y definir un perfil de personalidad. Esta evaluación se lleva a cabo a partir de los distintos componentes que conforman la personalidad. En este sentido, según Muñoz [75], los psicólogos utilizan cuatro técnicas básicas para medir la personalidad: la entrevista personal, la observación directa del comportamiento, los test objetivos y los test proyectivos.

La entrevista personal

La entrevista es una herramienta fundamental en el campo de la investigación psicológica, ya que permite obtener información valiosa acerca de la persona entrevistada. Se distinguen dos tipos de entrevista: la no estructurada, que permite una mayor libertad en la selección de temas y preguntas, y la estructurada, que se rige por un formato definido previamente. En investigaciones de personalidad, la entrevista estructurada es preferida por los investigadores al permitir una mayor precisión en la recolección de datos. Es común que la entrevista se utilice en conjunto con otras

técnicas, y puede emplearse en cualquier etapa del proceso de selección de personal. Además, puede funcionar tanto como complemento de otras pruebas psicológicas como por sí sola.

La observación directa del comportamiento

Observar cómo una persona actúa en ciertas situaciones puede proporcionar información sobre su personalidad y cómo es influenciada por el entorno. Sin embargo, la presencia de un observador puede influir en el comportamiento de la persona y complicar la interpretación de los resultados. Para complementar la observación, se puede utilizar la entrevista y la evaluación psicológica, adaptando los objetivos de observación a los de la entrevista y evaluación. Durante la observación se registran las reacciones ante estímulos que evalúan procesos psicológicos como la memoria, atención y lenguaje, lo que puede orientar las hipótesis de trabajo del observador. En la evaluación psicológica, se planifica la observación en tres etapas: antes, durante y después de la prueba, prestando mayor atención en pruebas de inteligencia y técnicas proyectivas.

Las pruebas objetivas

Las pruebas objetivas, también conocidas como inventarios, son cuestionarios estandarizados que recopilan las respuestas escritas de los sujetos, generalmente en formato de verdadero/falso o selección múltiple. Proporcionan información sobre diversos aspectos de la personalidad, incluyendo valores, necesidades, intereses, autoestima, problemas emocionales y patrones de comportamiento en situaciones específicas. Los psicólogos han utilizado estas pruebas para identificar cientos de rasgos de personalidad, convirtiéndolas en una herramienta valiosa en la evaluación psicológica.

Aunque las pruebas objetivas son consideradas más confiables y válidas que las proyectivas, su facilidad de administración puede dar la impresión equivocada de que pueden ser utilizadas por investigadores sin antecedentes en psicología y psicometría. Sin embargo, según McDonald y Edwards [76], la interpretación de los resultados y el análisis de sus implicaciones prácticas no son simples y requieren la capacitación adecuada de profesionales.

Las pruebas proyectivas

Se utilizan pruebas proyectivas de personalidad para estimular a las personas a expresar sus sentimientos y fantasías, y se basan en la suposición de que la personalidad está formada por deseos reprimidos y conflictos inconscientes. Estas pruebas pueden incluir asociación de palabras, dibujos o copias de diseños.

5) El modelo Big Five

En la actualidad, existen numerosos modelos y teorías de personalidad que ofrecen diferentes perspectivas sobre cómo abordar la personalidad de una persona. Entre ellas se encuentran Carl Jung's Psychological Types [77], Keirsey's Personality Types Theory [78], The Big Five Factors Personality Model [79] y Myers-Briggs Type Indicator (MBTI) [80], entre otras.

En este estudio, se ha optado por utilizar el modelo de personalidad conocido como Big Five debido a su amplio consenso en la comunidad científica de la Psicología y a su extenso uso en la literatura [81], [82]. Además, este modelo también ha sido utilizado en el campo de la IS [48]. Big Five es un modelo jerárquico de rasgos de personalidad que consta de cinco grandes factores, cada uno de los cuales representa características de personalidad a un nivel de abstracción más general.

Estas dimensiones suelen ser referidas de manera común como [24], [81], [83]:

- **E (Extraversion o extraversión):** Esta dimensión se caracteriza por una actitud activa y dinámica hacia el mundo social, e incluye rasgos como sociabilidad, asertividad, actividad y emociones positivas. Las personas extrovertidas se destacan por su habilidad para relacionarse fácilmente con los demás, disfrutan de la compañía de muchas personas, son impulsivos y les gusta correr riesgos. Además, suelen ser amantes de la aventura, las bromas y la variación, y se sienten muy comprometidos con el mundo exterior. Son personas entusiastas y orientadas a la acción, que cuentan con mucha energía, son asertivas y les gusta llamar la atención hacia sí mismas. Por otro lado, los introvertidos se caracterizan por ser más reservados y disfrutar de actividades que se practican en solitario, como la lectura. Son menos impulsivos, piensan antes de hablar, prefieren hacer planes a largo plazo y mantienen una vida más ordenada. Asimismo, son más tranquilos y retraídos y suelen tener un círculo social más pequeño, al que eligen cuidadosamente.
- **A (Agreeableness o amabilidad):** Esta dimensión se relaciona con la tendencia a tener conductas prosociales y altruistas. Se refiere a la habilidad para ser agradable con los demás, cooperar con ellos y mantener la armonía social. Las personas con altos niveles de amabilidad son empáticas, consideradas, generosas, amables y serviciales. Ven a los demás de forma positiva y tienden a creer que la mayoría de las personas son amistosas, honestas y dignas de confianza. Por otro lado, las personas con bajos niveles de amabilidad son menos empáticas y menos dispuestas a ayudar a los demás. Además, suelen ser más escépticas, desconfiadas y menos amistosas, prefiriendo competir en lugar de cooperar. Aquellos que obtienen puntuaciones extremadamente bajas en esta dimensión pueden ser manipuladores y antisociales.
- **C (Conscientiousness o meticulosidad):** Esta dimensión se refiere a la tendencia de una persona a ser responsable, confiable, minuciosa, cuidadosa y detallista en la realización de una tarea o actividad. Las personas altamente conscientes son organizadas y planificadas, establecen metas realistas y se esfuerzan por alcanzarlas. Son muy auto-disciplinadas y suelen seguir reglas y procedimientos de manera rigurosa. Además, son perseverantes y tienden a tener una fuerte ética de trabajo. Estas personas también tienden a ser muy fiables y comprometidas, lo que les permite ser líderes efectivos y trabajar en equipo de manera efectiva. Por otro lado, las personas con baja meticulosidad pueden parecer menos organizadas, menos confiables y menos comprometidas con el logro de sus objetivos. Tienden a ser más impulsivas, desorganizadas y menos perseverantes. También pueden tener dificultades para seguir reglas y procedimientos y pueden ser menos propensas a establecer metas realistas.
- **N (Neuroticism o inestabilidad emocional):** La estabilidad emocional (neuroticismo) es una dimensión que determina la capacidad de una persona para lidiar con el estrés y la frustración. Aquellos con baja estabilidad emocional no pueden tolerar condiciones de vida insatisfactorias y suelen reaccionar con intensas emociones negativas como ansiedad, tristeza, ira o culpa frente a los desafíos cotidianos. En cambio, las personas con alta estabilidad emocional tienen un mayor control sobre sus emociones y no se ven fácilmente afectadas por los problemas de la vida. Son emocionalmente maduros, pacientes, perseverantes y confiables, y poseen una capacidad para manejar sus emociones de manera flexible y controlada. Además, tienen una visión realista de la vida, no presentan síntomas neuróticos ni hipocondríacos, y pueden planear su vida y resistir sus impulsos con facilidad.
- **(Openness o apertura a nuevas experiencias):** Esta dimensión de la personalidad se refiere a la apertura mental, que abarca la originalidad, la creatividad, la imaginación, la curiosidad intelectual y la disposición a explorar nuevas ideas y experiencias. Las personas que obtienen

altas puntuaciones en esta dimensión son aquellos con mentes abiertas, flexibles y curiosas, que buscan constantemente nuevas experiencias y tienen una amplia gama de intereses. Son imaginativos, intuitivos, artísticos y disfrutan de la complejidad y la ambigüedad. Por otro lado, las personas con bajas puntuaciones en esta dimensión tienen mentes más cerradas y prefieren lo familiar y lo convencional. Son más resistentes al cambio y no disfrutan de las experiencias nuevas y diferentes. También pueden tener menos intereses y menos habilidades creativas. En resumen, la apertura mental se refiere a la disposición a explorar nuevos terrenos y aceptar nuevas ideas, mientras que las personas con baja apertura son más tradicionales y prefieren la comodidad de lo conocido.

La comprensión de las dimensiones de la personalidad y la combinación de valores en cada dimensión puede ayudar a identificar las habilidades y preferencias de las personas. Por lo tanto, el análisis de la personalidad puede ser una herramienta valiosa para aprovechar las habilidades de las personas y mejorar el desempeño en el trabajo y en otras áreas de la vida.

a) Instrumentos para medir las dimensiones del modelo Big Five

Si bien el modelo Big Five es un modelo descriptivo de la personalidad, los psicólogos han desarrollado varios tests y cuestionarios para evaluar cada una de las cinco dimensiones en los individuos. En la TABLA I se presenta un resumen de los más utilizados.

TABLA I
INSTRUMENTOS MÁS EMPLEADOS PARA MEDIR LAS DIMENSIONES DEL MODELO BIG FIVE

Instrumento	Descripción
Revised NEO Personality Inventory (NEO-PI-R) [79]	La denominación NEO hace referencia a las tres dimensiones originales que el inventario medía: Neuroticism, Extraversion y Openess. Este instrumento es uno de los más utilizados para medir los Cinco Grandes, ya que no solo evalúa las cinco dimensiones, sino que también analiza seis características específicas de cada una. Desarrollado por McRae y Costa, consta de 240 frases en las que el individuo debe indicar su acuerdo o desacuerdo a través de una escala del 1 al 5. A partir de estas frases se pueden identificar diferencias individuales y obtener 35 puntuaciones distintas.
Sixteen Personality Factor Questionnaire (16PF-5) [73]	Se trata de la versión más actualizada del clásico Test de Personalidad, basado en la teoría de los 16 Factores de Personalidad de Catell. Este instrumento se utiliza en práctica clínica y en selección de personal, debido a que brinda un informe detallado que proporciona una visión general de los Cinco Grandes, así como una interpretación más profunda de los 16 factores originales definidos por Catell.
Big Five Inventory (BFI) [84]	El cuestionario está compuesto por 44 ítems que constan de frases cortas y fáciles de comprender. Esta metodología mantiene las ventajas de los adjetivos, como su brevedad y simplicidad, mientras se evitan las desventajas de la ambigüedad, definiciones múltiples y la deseabilidad aparente. En algunos casos, se incluye información sobre el contexto o se aclaran los adjetivos para mejorar la comprensión. Los encuestados responden en una escala de 5 puntos, desde "totalmente en desacuerdo" hasta "totalmente de acuerdo". Estas escalas han demostrado una alta consistencia interna, confiabilidad en el re-testeo, una estructura factorial clara y una fuerte convergencia con versiones más extensas utilizadas para medir los Cinco Grandes, lo que favorece su uso.
Big Five Questionnaire (BFQ) [85]	Se trata de un cuestionario específico que deriva directamente de la teoría de los Cinco Grandes. El cuestionario consta de 132 elementos que la persona debe responder en una escala de múltiple elección. Este

TABLA I INSTRUMENTOS MÁS EMPLEADOS PARA MEDIR LAS DIMENSIONES DEL MODELO BIG FIVE	
Instrumento	Descripción
	instrumento es particularmente recomendado para los procesos de selección de personal, ya que se ha demostrado que las dimensiones evaluadas por este test están directamente relacionadas con las actividades habituales en la vida laboral.
Children Big Five Questionnaire (BFQ-C) [86]	Este es un cuestionario diseñado específicamente para niños y adolescentes, basado en el modelo Big Five. Consta de 65 ítems que se valoran individualmente en una escala de 5 opciones. Una ventaja de este cuestionario es que puede ser completado tanto por el propio niño como por sus padres o docentes, ya que los ítems pueden formularse en tercera persona para hacer referencia al sujeto de evaluación.

Fuente: esta investigación.

6) La personalidad y los grupos

La conformación de grupos es un proceso fundamental en la vida humana, ya que nos permite interactuar, compartir y aprender de otros individuos que comparten intereses, objetivos o necesidades similares [87]. Los antecedentes de la conformación de grupos se remontan a la prehistoria, cuando nuestros antepasados se unían en tribus para sobrevivir y protegerse mutuamente. Desde entonces, los grupos han evolucionado y se han especializado en diferentes ámbitos, como el trabajo, la educación, la religión, el deporte, entre otros. En la actualidad, la conformación de grupos sigue siendo una herramienta esencial para el desarrollo personal y social de las personas, y su importancia se evidencia en el impacto positivo que tienen en la vida de sus miembros [88].

La personalidad es un aspecto clave en la conformación y dinámica de los grupos, ya que influye en cómo los individuos se relacionan con los demás, cómo se perciben a sí mismos y cómo se adaptan a las normas y expectativas del grupo [89]. Por lo tanto, comprender la personalidad de los miembros de un grupo es esencial para entender su funcionamiento y su capacidad para alcanzar los objetivos propuestos. Los hallazgos de Whittingham [90] y Bradley et al. [91] sugieren que la personalidad de los estudiantes tiene un impacto significativo en su rendimiento de aprendizaje, así como en el papel crucial que juega la composición de la personalidad en el éxito del desempeño grupal. Además, la personalidad no es ajena en el campo de la IS, muchos estudios han demostrado que la personalidad puede tener un impacto significativo en el éxito del desarrollo de software, tanto a nivel individual como grupal [21], [22], [39], [92], [93]. Aunque la personalidad es un factor importante a tener en cuenta en la formación de grupos, es esencial considerar otros factores para asegurarse de que el grupo sea efectivo. Por ejemplo, la experiencia y las habilidades técnicas de los miembros del grupo también son críticas para el éxito del proyecto [32].

a) Tipos de grupos

Existen diversos tipos de grupos según su estructura y finalidad. A continuación, se describen algunos de los más comunes [94], [95]:

- Grupos formales: Son aquellos grupos que se crean de manera planificada y con una estructura definida, con objetivos claros y una autoridad establecida. Estos grupos se utilizan comúnmente en entornos laborales y académicos para llevar a cabo tareas específicas.
- Grupos informales: A diferencia de los grupos formales, los grupos informales no tienen una estructura ni objetivos específicos establecidos de antemano. Son formados por los propios

miembros en función de sus intereses y afinidades. Estos grupos pueden surgir en cualquier contexto social, como en el trabajo, en el colegio o en la comunidad.

- Grupos básicos: Los grupos básicos son aquellos que se forman en situaciones en las que las personas interactúan de manera regular y directa, como la familia, los amigos cercanos y los vecinos. Estos grupos suelen ser pequeños y tener una relación emocional más fuerte entre sus miembros.

Cada tipo de grupo tiene características y objetivos diferentes, y pueden ser utilizados en distintos contextos según las necesidades y objetivos específicos. Es importante tener en cuenta que los grupos pueden tener un gran impacto en la dinámica social, la toma de decisiones y el rendimiento individual y grupal, por lo que es esencial comprender su funcionamiento y potencial.

b) Tamaño de los grupos

Para lograr una mayor eficacia en el trabajo colaborativo, se recomienda que el tamaño del grupo se encuentre entre 2 y 6 personas, aunque esto puede variar dependiendo de diversos factores y preferencias individuales. Bean [96] argumenta que el tamaño más efectivo para grupos formales e informales de clase es de 5 personas, ya que si se supera este número, la experiencia se diluye. Por otro lado, los grupos de 4 tienden a dividirse en parejas y los de 3 suelen tener un miembro que no está completamente integrado. No obstante, en algunos casos, las parejas pueden trabajar mejor, especialmente en situaciones en las que se requiere una interrupción rápida de una lección magistral. En resumen, se considera que un tamaño ideal para un grupo de trabajo colaborativo estaría entre 3 y 5 personas, ya que permitiría tener suficiente diversidad de habilidades y perspectivas, mientras se mantiene un ambiente colaborativo y eficiente [96], [97]. Sin embargo, es importante tener en cuenta que el tamaño ideal del grupo puede variar según el contexto y los objetivos del proyecto.

c) Miembros de los grupos

Existen diversas formas de constituir grupos, sus miembros pueden ser asignados al azar, seleccionados por los estudiantes, establecerlos a criterio del docente o utilizar herramientas computacionales para su conformación. La pertenencia a un grupo puede basarse en intereses, habilidades, actitudes u otras características, y estos pueden ser heterogéneos o homogéneos. En general, la investigación respalda la formación de grupos heterogéneos, ya que esto permite que los estudiantes interactúen con individuos que poseen diversas ideas, antecedentes y experiencias, lo cual tiene un valor importante en la productividad y a la adaptación a tareas multidimensionales [78]–[80]. Sin embargo, los grupos heterogéneos presentan algunas desventajas. Los miembros pueden sentirse incómodos ante la diversidad de opiniones, antecedentes, experiencias y las posibles tensiones que se puedan derivar de los desacuerdos, dificultando la colaboración y la comunicación en el grupo[101]–[103].

Por otro lado, los grupos homogéneos son aquellos grupos en los que sus miembros comparten características, habilidades, intereses, experiencias y/o valores similares. Estas similitudes pueden ser en términos de edad, género, educación, personalidad, antecedentes culturales, entre otros factores [104], [105]. Los grupos homogéneos tienden a tener una mayor cohesión, ya que sus miembros comparten intereses y valores comunes. Esto puede mejorar la comunicación y el trabajo en equipo, lo que a su vez puede aumentar la eficiencia y eficacia del grupo [28][106]. Sanz-Martínez et al. [107] reveló que considerar la homogeneidad en la participación de los estudiantes como un criterio para formar grupos condujo a mejores resultados en términos de rendimiento

grupal, interacciones grupales y satisfacción de los estudiantes. Así mismo, Beane et al. [108] llegó a la conclusión de que los estudiantes de bajo rendimiento se benefician de un agrupamiento homogéneo. Sin embargo, la desventaja de estos grupos es que puede limitar la diversidad de ideas y perspectivas dentro del grupo. Los miembros del grupo que comparten características similares, como antecedentes culturales, educativos o de personalidad, pueden estar más inclinados a pensar de manera similar y a evitar el desafío de las ideas de otros miembros del grupo. Es importante tener en cuenta que cada tipo de grupo tiene sus ventajas y desventajas, y que la elección dependerá del contexto específico y los objetivos del proyecto o actividad en cuestión.

7) Calidad del software

En esta sección, se abordarán varios aspectos clave relacionados con los atributos de calidad del software. En primer lugar, se describirá de forma breve el concepto de la calidad de software, seguido de una explicación resumida de los estándares y modelos de calidad del software. Por último, se hablará sobre los atributos de mantenibilidad y adecuación funcional, que son fundamentales para la evaluación del producto software en el estudio que se está llevando a cabo.

a) Concepto de calidad del software

En términos generales, la calidad del software se refiere a las características y propiedades que lo hacen confiable, eficiente, seguro, mantenible y fácil de usar. Es decir, se considera de calidad aquel software que cumple con los requisitos del cliente o usuario final, y que se desarrolla siguiendo buenas prácticas y estándares de la industria [2], [109]. Según Bourque et al. [99], los requisitos del software definen los atributos de calidad necesarios y afectan los métodos de medición y criterios de aceptación para evaluar el grado en que el software y su documentación alcanzan los niveles de calidad deseados.

Desde la década de 1970, la calidad del software se ha vuelto un tema crítico para la industria de la tecnología de la información debido a los problemas de confiabilidad y estabilidad presentados por los primeros sistemas de software [2]. Desde entonces, se han desarrollado numerosas metodologías y estándares para garantizar la calidad del software, tales como ISO 9001, CMMI, TSP/PSP, Agile, entre otros [110]. La importancia de la calidad del software radica en las consecuencias que puede tener tanto para las empresas como para los usuarios finales. Los errores y fallas en el software pueden generar pérdidas financieras, dañar la reputación de la empresa y afectar la seguridad de los usuarios finales. Por ejemplo, un fallo en un software de control de tráfico aéreo podría poner en riesgo la vida de las personas, mientras que un error en un software bancario podría causar la pérdida de datos sensibles de los clientes.

Actualmente, la calidad del software sigue siendo un aspecto vital para la satisfacción del cliente, el rendimiento de la empresa y la seguridad de los usuarios finales. Además, la calidad del software es esencial para la innovación y el progreso en la tecnología de la información, ya que un software de alta calidad puede generar mayores oportunidades y beneficios para las empresas y los usuarios. Por lo tanto, es importante que las empresas y desarrolladores de software se enfoquen en garantizar y mejorar continuamente la calidad de su software para mantenerse competitivos en un mercado cada vez más exigente y cambiante.

Es fundamental tener en cuenta los estándares y modelos para crear software de calidad, ya que estos proporcionan una guía estructurada y un conjunto de buenas prácticas para el proceso de desarrollo de software. Estos estándares y modelos establecen requisitos específicos para el diseño, la codificación, las pruebas y la documentación del software, lo que garantiza que el software

cumpla con los criterios de calidad y funcionalidad necesarios para su uso efectivo [111]. Además, estos estándares y modelos también proporcionan un marco para la evaluación y mejora continua del proceso de desarrollo de software, lo que ayuda a identificar y corregir problemas en el proceso [112]. En general, seguir los estándares y modelos de desarrollo de software contribuye a la creación de software de calidad, que cumple con las necesidades del usuario y es fácil de mantener y mejorar en el futuro.

b) Estándares y modelos de calidad de software

Los modelos de calidad del producto software y los procesos asociados son fundamentales en el desarrollo de software de alta calidad y confiabilidad. Estos modelos definen un conjunto de estándares, prácticas y procedimientos que se deben seguir para garantizar que el software cumpla con los requisitos de calidad establecidos y las necesidades de los usuarios [2], [111].

Un modelo de calidad del producto software establece un conjunto de criterios y métricas que se utilizan para evaluar la calidad del software [113]. Estos criterios pueden incluir aspectos como la funcionalidad, el rendimiento, la usabilidad, facilidad de mantenimiento, entre otros. Al utilizar un modelo de calidad del producto, los desarrolladores de software pueden establecer objetivos claros y medibles para la calidad del software y trabajar para alcanzarlos [111], [113]. Por otro lado, los procesos de desarrollo de software son igualmente importantes. Estos procesos establecen un marco para la planificación, diseño, implementación, prueba y mantenimiento del software. Un proceso de desarrollo de software bien definido y documentado ayuda a garantizar que los proyectos se entreguen a tiempo y dentro del presupuesto, mientras se mantiene la calidad del producto [112].

Al seguir un modelo de calidad del producto y un proceso de desarrollo de software, los desarrolladores pueden minimizar los errores y defectos en el software, lo que a su vez reduce los costos de mantenimiento y mejora la satisfacción del usuario final. Además, seguir un modelo de calidad del producto y un proceso de desarrollo de software puede ayudar a garantizar que el software cumpla con las regulaciones y estándares de la industria.

c) Modelos de calidad del proceso software

Algunos de los modelos y estándares más comunes para evaluar la calidad del proceso de software son:

CMMI (Capability Maturity Model Integration). Es un modelo de mejora de procesos que proporciona un enfoque sistemático y estructurado para mejorar la calidad y la eficiencia de los procesos en una organización. CMMI se desarrolló originalmente en el Software Engineering Institute (SEI) de la Universidad Carnegie Mellon en Estados Unidos y se ha convertido en un marco ampliamente utilizado para la mejora de procesos en organizaciones de diversos sectores y tamaños [114].

CMMI proporciona una serie de prácticas y procesos recomendados que las organizaciones pueden adoptar y mejorar para lograr niveles crecientes de madurez en sus procesos. Los niveles de madurez van del 0 al 5 siendo el 0 el mínimo nivel (empresa inmadura) y el 5 el máximo (empresa en continuo estado de mejora e innovación en desarrollo software). Cada nivel se compone de un conjunto de áreas de proceso que cubren diferentes aspectos del ciclo de vida del software, desde la gestión de proyectos hasta la gestión de configuración y la garantía de calidad [115].

CMMI se puede utilizar para evaluar la madurez de los procesos de una organización y para identificar áreas de mejora y oportunidades de mejora. También puede ayudar a las organizaciones a establecer objetivos claros de mejora de procesos y a medir el progreso hacia estos objetivos a lo largo del tiempo. CMMI es ampliamente utilizado en el sector de la tecnología de la información y las comunicaciones, así como en otros sectores, como la ingeniería, la manufactura y la defensa [113].

ISO/IEC 15504. También conocido como SPICE, es un estándar internacional que se utiliza para evaluar y mejorar los procesos de desarrollo de software. SPICE se basa en un modelo de procesos de ciclo de vida que consta de cinco categorías de procesos y seis niveles de capacidad. La evaluación SPICE se realiza mediante un proceso de evaluación formal que implica evaluadores externos, mientras que la mejora de procesos de software se logra identificando los procesos que necesitan mejorar y estableciendo objetivos de mejora para mejorar la capacidad de los procesos de desarrollo de software y la calidad del software producido [116]. En general, SPICE es una herramienta valiosa para las organizaciones que desean mejorar sus procesos de desarrollo de software y, por lo tanto, aumentar la calidad y la eficiencia de sus productos de software [116].

ISO/IEC 12207. Es un estándar internacional para el ciclo de vida del software. Proporciona un marco de referencia para el desarrollo, mantenimiento y retirada de software, y es aplicable a cualquier tipo de software. El estándar se divide en tres secciones [117]:

- La sección de proceso describe los procesos que se utilizan durante el ciclo de vida del software. Estos procesos se dividen en dos categorías: procesos principales y procesos de soporte. Los procesos principales se utilizan para desarrollar y mantener el software, mientras que los procesos de soporte se utilizan para proporcionar apoyo y asegurar la calidad del software.
- La sección de ciclo de vida describe las fases y actividades que se llevan a cabo durante el ciclo de vida del software. Estas fases incluyen la planificación, el análisis de requisitos, el diseño, la implementación, las pruebas, la instalación y la operación y mantenimiento. Cada fase se compone de una serie de actividades que deben realizarse para completar la fase.
- La sección de gestión describe cómo se gestiona el ciclo de vida del software. Esto incluye la gestión de la configuración, la gestión de los riesgos, la gestión de la calidad y la gestión de proyectos. También se describe cómo se gestiona la documentación del software y cómo se gestionan los cambios y las revisiones del software.

Modelo IEEE 1061. es un estándar de prácticas recomendadas para la evaluación de software. Este modelo proporciona una guía para evaluar la calidad del software, teniendo en cuenta varios factores, como la funcionalidad, la confiabilidad, la usabilidad, la eficiencia y la mantenibilidad.

El modelo IEEE 1061 consta de cuatro fases principales [118]:

- Planificación de la evaluación: en esta fase se establecen los objetivos de la evaluación y se definen los criterios para medir el rendimiento del software. También se determinan los recursos necesarios para realizar la evaluación.
- Preparación de la evaluación: en esta fase se preparan los casos de prueba y se realiza la recopilación de datos necesarios para llevar a cabo la evaluación. También se establecen los procedimientos de evaluación y se planifican los métodos para la recolección de datos.
- Realización de la evaluación: en esta fase se lleva a cabo la evaluación propiamente dicha. Se ejecutan los casos de prueba y se recopilan los datos necesarios para evaluar la calidad del

software. También se analizan los resultados de la evaluación y se generan informes para comunicar los hallazgos.

- Informe y seguimiento de la evaluación: en esta fase se elabora un informe de evaluación detallado que incluye los resultados de la evaluación y las recomendaciones para mejorar la calidad del software. También se realiza un seguimiento para garantizar que se implementen las recomendaciones y que se mejore continuamente la calidad del software.

d) Modelos de calidad del producto software

Algunos de los modelos más comunes para evaluar la calidad del producto de software son:

ISO/IEC 9126. Es un estándar internacional que define un marco de referencia para la evaluación de la calidad del software. Este modelo proporciona un conjunto de características que se utilizan para evaluar la calidad del software, así como un conjunto de métricas para medir la calidad en cada una de estas características [113], [119].

El modelo ISO/IEC 9126 se divide en dos partes principales: una para la evaluación de la calidad interna del software y otra para la evaluación de la calidad externa del software.

La calidad interna del software se refiere a la calidad del código fuente y del diseño del software, y se mide a través de las siguientes características [119]:

- Funcionalidad: se refiere a la capacidad del software para cumplir con los requisitos funcionales.
- Fiabilidad: se refiere a la capacidad del software para funcionar sin errores y para mantener un nivel de rendimiento constante.
- Usabilidad: se refiere a la facilidad de uso del software para los usuarios finales.
- Eficiencia: se refiere al rendimiento y la eficiencia del software, es decir, la rapidez y el uso eficiente de los recursos.
- Mantenibilidad: se refiere a la facilidad con que el software puede ser modificado y mantenido.

La calidad externa del software se refiere a la calidad percibida por los usuarios y se mide a través de las siguientes características [119]:

- Funcionalidad: se refiere a la capacidad del software para cumplir con las necesidades y expectativas de los usuarios.
- Fiabilidad: se refiere a la capacidad del software para funcionar sin errores y para mantener un nivel de rendimiento constante en un entorno de usuario.
- Usabilidad: se refiere a la facilidad de uso del software para los usuarios finales.
- Eficiencia: se refiere al rendimiento y la eficiencia del software, es decir, la rapidez y el uso eficiente de los recursos.
- Mantenibilidad: se refiere a la facilidad con que el software puede ser modificado y mantenido para satisfacer las necesidades futuras.

ISO/IEC 25010. Es un estándar internacional que define un marco para la evaluación de la calidad del software y los sistemas de información. Este modelo proporciona un conjunto de características para evaluar la calidad del producto de software, y se utiliza para identificar y medir la calidad de un software o sistema de información [120]. El modelo ISO/IEC 25010 reemplaza al modelo ISO/IEC 9126, que se centraba en la calidad del software, y lo amplía para incluir la evaluación de la calidad de los sistemas de información. El modelo ISO/IEC 25010 se basa en el modelo ISO/IEC

9126 y en otros estándares internacionales relevantes, pero proporciona una estructura de evaluación más detallada y completa [113], [120].

La norma proporciona un marco de trabajo para evaluar la calidad de los sistemas de información y el software desde diferentes perspectivas, incluyendo la calidad en uso y la calidad del producto. El modelo de calidad en uso se enfoca en el resultado de la interacción entre el usuario y el producto en un contexto específico de uso. Este modelo está compuesto por cinco características principales [121]:

- Adecuación funcional: Esta característica se refiere a la capacidad del producto para proporcionar las funciones necesarias para satisfacer los requisitos del usuario en un contexto específico de uso.
- Eficiencia del desempeño: Esta característica se refiere a la capacidad del producto para proporcionar un rendimiento aceptable en términos de tiempo de respuesta, tasa de procesamiento y uso de recursos del sistema en un contexto específico de uso.
- Compatibilidad: Esta característica se refiere a la capacidad del producto para ser compatible con otros sistemas y entornos en un contexto específico de uso.
- Usabilidad: Esta característica se refiere a la facilidad de uso del producto, incluyendo la facilidad de aprendizaje, la facilidad de uso y la satisfacción del usuario.
- Seguridad: Esta característica se refiere a la capacidad del producto para proteger la información y los recursos del sistema en un contexto específico de uso.

El modelo de calidad del producto se enfoca en las propiedades estáticas y dinámicas del software y el sistema informático. Este modelo está compuesto por ocho características principales [121]:

- Funcionalidad: Esta característica se refiere a la capacidad del software para proporcionar las funciones necesarias para satisfacer los requisitos del usuario.
- Fiabilidad: Esta característica se refiere a la capacidad del software para mantener su nivel de rendimiento durante un período de tiempo determinado.
- Usabilidad: Esta característica se refiere a la facilidad de uso del software, incluyendo la facilidad de aprendizaje, la facilidad de uso y la satisfacción del usuario.
- Eficiencia: Esta característica se refiere a la capacidad del software para utilizar los recursos del sistema de manera eficiente.
- Mantenibilidad: Esta característica se refiere a la capacidad del software para ser modificado y mantenido de manera eficiente.
- Portabilidad: Esta característica se refiere a la capacidad del software para funcionar en diferentes entornos y plataformas.
- Compatibilidad: Esta característica se refiere a la capacidad del software para ser compatible con otros sistemas y entornos.
- Seguridad: Esta característica se refiere a la capacidad del software para proteger la información y los recursos del sistema.

Cada una de estas características se subdivide en subcaracterísticas específicas para ayudar en la evaluación de la calidad.

En resumen, ISO/IEC 25010 proporciona un marco de trabajo detallado para evaluar la calidad de los sistemas de información y el software desde diferentes perspectivas. Este modelo de calidad

ayuda a los usuarios, desarrolladores y evaluadores a entender mejor la calidad de un producto de software o sistema de información y proporciona una guía para mejorar la calidad.

Modelo McCall. Es un modelo de calidad de software que fue desarrollado por John McCall, James Richards y Glenn Walters en 1977. Este modelo fue uno de los primeros intentos de definir las características de calidad del software y proporcionar un marco para la evaluación de la calidad del software [120].

El modelo McCall se divide en tres categorías principales de calidad de software [120]:

- Características operativas: estas características se refieren a cómo el software funciona en su entorno operativo. Las características operativas incluyen la corrección, la eficiencia, la confiabilidad, la integridad y la usabilidad.
- Características de revisión: estas características se refieren a la facilidad de mantenimiento y revisión del software. Las características de revisión incluyen la facilidad de mantenimiento, la facilidad de prueba, la flexibilidad y la portabilidad.
- Características de transición: estas características se refieren a cómo el software se mueve de su entorno de desarrollo a su entorno de producción. Las características de transición incluyen la capacidad de instalación, la capacidad de adaptación y la capacidad de conversión.

Cada una de estas categorías de calidad del software se divide en subcaracterísticas específicas para ayudar en la evaluación de la calidad del software. Por ejemplo, la categoría de características operativas incluye la corrección, que se divide en subcaracterísticas como la precisión, la consistencia y la recuperación de errores.

El modelo McCall fue muy influyente en su época y ayudó a establecer las bases para futuros modelos de calidad de software. Aunque el modelo McCall es bastante antiguo, muchas de las características de calidad del software que define siguen siendo relevantes en la actualidad.

En la Fig. 2 se presenta un mapa mental que muestra las tendencias en las métricas de calidad del producto de software, resultado de un estudio de mapeo sistemático realizado por Colakoglu et al. [113]. En este estudio analizaron el mapa de tendencias entre los años 2009 y 2019, el conocimiento existente en esta área, las herramientas de medición utilizadas, los temas que se identificaron como áreas de desarrollo potencial y la concordancia entre los documentos de conferencias, artículos y modelos de calidad internacionalmente reconocidos.

Fig. 2. Tendencia de las métricas de calidad del producto de software.

Fuente: Tomado de Colakoglu et al. [113].

El presente estudio se ha enfocado en los atributos de mantenibilidad y adecuación funcional debido a su importancia en la evaluación de la calidad del software. De acuerdo con diversos estudios e investigaciones en el campo de la IS, estos atributos son cruciales para garantizar que el software sea útil y valioso para los usuarios a largo plazo [49], [54], [113], [122]. Además, estos atributos están altamente relacionados, los desarrolladores de software tienen una gran responsabilidad en asegurar que el software cumpla con estos atributos [123],[124].

Aunque la mantenibilidad y adecuación funcional son esenciales para evaluar la calidad del software, es crucial destacar que existen otros atributos que también merecen ser considerados. Entre ellos, se encuentran la eficiencia, seguridad, usabilidad, entre otros factores críticos que afectan tanto la calidad del software como la satisfacción del usuario. Evaluar cada uno de estos atributos puede ser una tarea desafiante y que requiere mucho tiempo y esfuerzo. Por lo tanto, es importante encontrar un equilibrio adecuado entre la profundidad del análisis y la eficiencia en la evaluación de la calidad del software. En este estudio, reconocemos la importancia de estos atributos y alentamos investigaciones futuras que se centren en su evaluación.

A continuación, se describe los atributos de mantenibilidad y adecuación funcional.

8) Mantenibilidad del software

Los esfuerzos realizados durante el proceso de desarrollo de software tienen como objetivo entregar un producto que cumpla con los requisitos del usuario. Sin embargo, una vez que el software está en funcionamiento, es probable que surjan nuevos requisitos y se presenten problemas que deben ser abordados. Estos pueden incluir la detección de defectos, cambios en el entorno operativo y la aparición de nuevas necesidades por parte del usuario. Aunque la fase de mantenimiento del ciclo de vida del software comienza después de la garantía o el soporte posterior a la implementación, las actividades de mantenimiento pueden ser necesarias desde etapas anteriores del desarrollo. Por lo tanto, es importante reconocer que el mantenimiento del software es un proceso continuo que implica identificar, corregir y mejorar el software en respuesta a los cambios y necesidades del usuario a lo largo del tiempo [112].

Según la guía de Swebook [112], el mantenimiento del software abarca todas las actividades necesarias para proporcionar soporte rentable al software, tanto antes como después de su entrega. Las actividades previas a la entrega incluyen la planificación de las operaciones posteriores, la evaluación de la capacidad de mantenimiento y la logística para la transición del software. Por otro lado, las actividades posteriores a la entrega involucran la modificación del software, la capacitación del usuario y la interacción con un equipo de soporte técnico en caso de requerirse. Es decir, el mantenimiento del software no se limita solo a la corrección de errores, sino que también incluye la adaptación del software a las necesidades cambiantes del usuario y su entorno, lo que garantiza su utilidad a largo plazo.

El mantenimiento del software es una de las fases más costosas y que requiere recursos del proceso de desarrollo de software, la evaluación de la mantenibilidad es un componente esencial del ciclo de vida del desarrollo de software moderno [49]. Aunque no existe un consenso general sobre la incidencia del mantenimiento en los costos de desarrollo, diversos estudios y estimaciones sugieren que puede llegar a representar entre el 60% y el 90% del total invertido a lo largo de la vida útil de un producto de software [125].

En el contexto del software, existen cuatro tipos de mantenimiento que se pueden realizar en un sistema [126]:

- Mantenimiento correctivo: Se realiza para corregir errores o fallas en el software después de que ha sido implementado. El mantenimiento correctivo se utiliza para solucionar problemas que pueden haber surgido durante el desarrollo del software o después de su implementación en producción. Este tipo de mantenimiento suele ser reactivo y se realiza cuando se detectan errores en el sistema o cuando los usuarios informan de problemas.
- Mantenimiento preventivo: Se realiza para evitar futuros problemas y mejorar la calidad del software. Este tipo de mantenimiento se realiza antes de que ocurra un problema y puede incluir la identificación de problemas potenciales y la aplicación de correcciones y mejoras para prevenirlos. El mantenimiento preventivo puede incluir actividades como la revisión de código, pruebas de rendimiento y actualizaciones de seguridad.
- Mantenimiento adaptativo: Se realiza para adaptar el software a los cambios en el entorno en el que se encuentra. Este tipo de mantenimiento se lleva a cabo para hacer frente a cambios en los requisitos del sistema o para adaptar el software a nuevas tecnologías. Por ejemplo, si el sistema operativo subyacente se actualiza, puede ser necesario realizar cambios en el software para que siga funcionando correctamente.
- Mantenimiento Perfectivo: Se enfoca en mejorar la calidad del software existente sin agregar nuevas funcionalidades. Este tipo de mantenimiento se utiliza para optimizar el software

existente, eliminar redundancias, mejorar la legibilidad y facilidad de mantenimiento del código, y en general para mejorar la calidad del software. Por ejemplo, se puede realizar un mantenimiento perfectivo para mejorar el rendimiento del software existente, reducir la complejidad del código o mejorar la seguridad.

Existen diversas formas de medir la mantenibilidad de los elementos de software sujetos o por someterse a mantenimiento, tales como el código, documentos de usuario y documentos de análisis o diseño. Las métricas de software se pueden clasificar en tres categorías [109],[127]:

- **Métricas del producto.** Describen las características del producto que de alguna forma determinan la mantenibilidad, como el tamaño, complejidad o características del diseño.
- **Métricas del proceso**. Pueden utilizarse para mejorar el desarrollo y mantenibilidad del software, como la eficacia de eliminar defectos durante el desarrollo, el patrón en el que aparecen los defectos durante las pruebas o el tiempo fijo de respuesta del proceso.
- **Métricas del proyecto.** Describen las características y ejecución del proyecto, tales como el número de desarrolladores, el patrón de staffing en el ciclo de vida, coste, planificación y productividad del software. En general, las métricas orientadas a objetos son de gran importancia para evaluar la mantenibilidad del software y mejorar su calidad.

Aunque las anteriores categorías de métricas son importantes, debe señalarse que, en este trabajo se evalúa la mantenibilidad del producto software centrándose específicamente en medir y evaluar la calidad del código fuente. Es importante destacar que el código fuente es el componente fundamental del software y su calidad es un factor crítico para determinar la facilidad con la que se pueden realizar modificaciones, actualizaciones, mejoras y correcciones en el software [127]. De hecho, existe una fuerte correlación entre la complejidad lógica del código fuente y la mantenibilidad del software resultante, tal como se ha demostrado en estudios previos [124], [128]–[130]. Además, el código fuente es el componente que está más expuesto a errores y fallos, por lo que es importante prestar especial atención a su mantenibilidad para garantizar el correcto funcionamiento del software a largo plazo.

a) Métricas de mantenibilidad del producto software

Las métricas de mantenibilidad del software son indicadores cuantitativos que permiten evaluar la facilidad con la que se puede mantener, evolucionar y mejorar un producto software a lo largo de su ciclo de vida [125]. Estas métricas se basan en el análisis de diversas características del software, como su complejidad, cohesión y acoplamiento, legibilidad del código, cumplimiento de estándares y buenas prácticas de programación, entre otras [49]. La medición y análisis de estas métricas permite obtener una visión más clara y objetiva de la calidad del software, y tomar decisiones informadas sobre cómo mejorar su mantenibilidad y gestionar los costos asociados al mantenimiento y evolución del mismo.

En lo que respecta a la evaluación de la calidad de un producto de software, es importante tener en cuenta las métricas del producto, las cuales se pueden dividir en dos clases [131]:

- **Métricas dinámicas:** estas métricas se recopilan mediante mediciones realizadas en un programa mientras está en ejecución. Se pueden obtener durante las pruebas del sistema o después de que el sistema está en uso. Un ejemplo de este tipo de métrica es el número de reportes de errores o el tiempo necesario para completar una tarea específica.

- **Métricas estáticas:** estas métricas se obtienen mediante mediciones realizadas en representaciones del sistema, como el diseño, el programa o la documentación. Algunos ejemplos de estas mediciones incluyen el tamaño del código y la longitud promedio de los identificadores utilizados.

Estos tipos de métrica se relacionan con diferentes atributos de calidad. Por un lado, las métricas dinámicas permiten valorar la eficiencia y fiabilidad de un programa en tiempo de ejecución. Por otro lado, las métricas estáticas se enfocan en evaluar la complejidad, comprensibilidad y mantenibilidad del sistema de software o de los componentes del sistema. Estas últimas son especialmente importantes ya que permiten identificar problemas en el código fuente y tomar medidas para mejorar la calidad del software en términos de facilidad de mantenimiento y evolución a largo plazo [131].

Por lo tanto, abordar las métricas internas relacionadas al código fuente y al paradigma de programación utilizado, como la programación orientado a objetos (POO), se puede cuantificar la mantenibilidad realizando mediciones específicas al código fuente del sistema a través de las métricas del producto software [49], [130]–[133]. Al utilizar POO se pueden diseñar sistemas de software modulares y reutilizables, lo que facilita la mantención y evolución del software a largo plazo [49], [134]. Al dividir el software en pequeñas piezas independientes, la POO permite realizar cambios y actualizaciones en una parte del sistema sin afectar a las demás, lo que reduce el riesgo de errores y fallos en el software [131], [134]–[137].

Por otro lado, en la evaluación de la mantenibilidad de un sistema OO se consideran cinco factores principales: complejidad, clase, acoplamiento, herencia y número de hijos [49]. Estos factores han sido elegidos debido a que son los factores más relevantes que afectan la complejidad del diseño y tienen un mayor impacto en la mantenibilidad. En la TABLA II se presentan breves descripciones de cada uno de estos factores [49]:

TABLA II
FACTORES PRINCIPALES QUE AFECTAN LA MANTENIBILIDAD DE UN SISTEMA OO

Complejidad	La complejidad del software se refiere a la dificultad que conlleva preservar, modificar y comprender el software. Esta característica es importante para evaluar la calidad de un programa OO, ya que una alta complejidad puede hacer que el software sea difícil de mantener y mejorar.
Clase	Una clase es una unidad fundamental en la programación orientada a objetos, y se puede definir como un conjunto de objetos que comparten los mismos atributos, métodos y relaciones. Las clases son una forma eficiente de estructurar y organizar el código en POO, y son una parte integral de la evaluación de la calidad del software OO.
Acoplamiento	El acoplamiento se refiere a la interdependencia entre diferentes componentes o funciones de un programa. Es una medida de la conexión entre los módulos de un software, y una alta interconexión puede hacer que el software sea difícil de mantener y modificar. Por lo tanto, el acoplamiento es una consideración importante al evaluar la calidad del software OO.
Herencia	La herencia es un concepto fundamental en la programación orientada a objetos, en el cual una clase puede heredar propiedades y métodos de otra clase. La herencia permite la reutilización de código y una mayor eficiencia en el desarrollo de software. Es un aspecto importante en la evaluación de la calidad del software OO, ya que una jerarquía de clases mal diseñada puede afectar negativamente la mantenibilidad del software.

Número de hijos	El número de hijos se refiere al número de subclases que están subordinadas a una clase en la jerarquía de herencia. Es una medida de la influencia potencial de una clase en el diseño y la estructura del sistema. El número de hijos es un factor importante a considerar al evaluar la calidad del software OO, ya que una clase con un gran número de hijos puede tener un impacto significativo en la complejidad y mantenibilidad del software.

Fuente: esta investigación.

b) Métricas Orientada a objetos

Las métricas OO son un conjunto de medidas cuantitativas que se utilizan para evaluar y mejorar la calidad del software OO [138]. Se utilizan para medir la complejidad del software, la calidad del diseño, la eficiencia y la mantenibilidad del software. Estas métricas se aplican a la arquitectura, el diseño, la implementación y la evolución del software OO [49], [130], [139]. De acuerdo con una revisión exhaustiva de la literatura realizada por Varela et al. [140], se identificaron cerca de 300 métricas de código fuente, siendo la programación orientada a objetos el paradigma más comúnmente estudiado. Esto refleja la importancia de las métricas OO en la evaluación de la calidad del software en dicho paradigma, lo que permite identificar posibles problemas y áreas de mejora en el diseño y la implementación del software. En resumen, las métricas OO son una herramienta crucial para el mantenimiento y evolución del software OO de alta calidad.

En el ámbito de la IS, existen varias métricas OO que se utilizan para medir distintos aspectos de los sistemas de software. Entre las suites de métricas más conocidas se encuentran Quality Metrics for Object-Oriented Design (QMOOD), Object-Oriented Metrics Suite (MOOD), Object-Oriented Software Engineering (OOSE), Chidamber y Kemerer (CK), entre otras. Aunque cada métrica se enfoca en diferentes aspectos, se considera que las métricas CK son las más estudiadas y relevantes en la industria y en la investigación académica [23], [124], [130], [138], [141]–[144].

Chidamber y Kemerer (CK)

Las métricas CK fueron desarrolladas por Shyam R. Chidamber y Chris F. Kemerer en la década de 1990. En su artículo "A Metrics Suite for Object Oriented Design" publicado en 1994 [129], presentaron una suite de métricas para evaluar la complejidad del diseño OO y la calidad del código. Esta suite de métricas se convirtió en una de las más utilizadas y estudiadas en el campo de la IS. Las métricas CK se enfocan en seis aspectos clave: WMC (weighted methods per class o métodos ponderados por clase), DIT (depth of inheritance tree o profundidad del árbol de herencia), NOC (number of children o número de hijos), CBO (coupling between objects o acoplamiento entre objetos), RFC (response for a class o respuesta para una clase) y LCOM (lack of cohesion in methods o falta de cohesión en métodos). Estas métricas son consideradas importantes para evaluar la mantenibilidad, la calidad y la complejidad de los sistemas OO [23], [49], [130], [144]. Desde su introducción, las métricas CK han sido objeto de numerosos estudios e investigaciones para evaluar su eficacia en la evaluación de la calidad del código y la complejidad del diseño [140]. Aunque ha habido ciertas críticas y limitaciones de estas métricas, continúan siendo ampliamente utilizadas [23], [144]–[146].

A continuación se detalla cada una de las métricas CK [49], [129]:

WMC (weighted methods per class o métodos ponderados por clase)

Es una medida de la complejidad de una clase y se calcula como la suma ponderada de todos los métodos definidos en la clase. Esta métrica también proporciona una estimación del tiempo y

esfuerzo requeridos para desarrollar y mantener la clase. Un valor alto de WMC indica una mayor complejidad y, por lo tanto, una menor capacidad de mantenimiento. Su definición matemática está definida de la siguiente manera:

$$WMC = \sum_{i=1}^{n} c_i$$

Donde C_i es el número de métodos definidos en la clase "i", por lo tanto, esta fórmula calcula la suma de todos los métodos definidos en una clase.

Puntos de vista:

- La cantidad de métodos y la complejidad de los métodos involucrados es un predictor de cuánto tiempo y esfuerzo se requiere para desarrollar y mantener la clase.
- Cuanto mayor sea el número de métodos en una clase, mayor será el impacto potencial en los objetos, ya que estos heredarán todos los métodos definidos en la clase.
- Es probable que las clases con un gran número de métodos sean más específicas de la aplicación, lo que limita la posibilidad de reutilización.

Una clase con un valor alto de WMC puede ser más difícil de mantener y comprender, ya que hay más métodos definidos en ella y es más compleja en términos de funcionalidad. Por lo tanto, WMC se utiliza para identificar clases que pueden requerir una mayor atención en términos de mantenimiento y refactorización para mejorar la calidad del código y la eficiencia del software.

DIT (depth of inheritance tree o profundidad del árbol de herencia)

Se define como la longitud máxima del camino desde la clase raíz de la jerarquía hasta la clase en cuestión en la jerarquía. En otras palabras, DIT mide la distancia de una clase desde la raíz de la jerarquía de clases.

Puntos de vista:

- Cuanto más profunda esté una clase en la jerarquía, mayor será el número de métodos que es probable que herede, lo que hace que sea más complejo predecir su comportamiento.
- Los árboles más profundos constituyen una mayor complejidad de diseño, ya que se involucran más métodos y clases.
- Cuanto más profunda sea una clase particular en la jerarquía, más métodos y clases estarán involucrados. mayor es la reutilización potencial de los métodos heredados.

DIT puede ser útil para identificar problemas de diseño en una jerarquía de clases. Un valor alto en esta métrica puede indicar que la jerarquía es demasiado profunda y compleja, lo que puede dificultar el mantenimiento y la comprensión del código. Además, una jerarquía de herencia profunda puede aumentar el acoplamiento entre clases y disminuir la cohesión, lo que puede afectar negativamente la calidad del software y una mayor probabilidad de fallos en el software. Por otro lado, puntajes muy bajos puede indicar que la jerarquía es demasiado plana y que hay una falta de abstracción en el diseño.

NOC (number of children o número de hijos)

Es una métrica de complejidad de software que se utiliza para medir el número de subclases directas que tiene una clase en una jerarquía de clases. En otras palabras, NOC mide el número de clases que heredan directamente de una clase.

Puntos de vista:

- A mayor número de hijos, mayor reutilización, ya que la herencia es una forma de reutilización.
- Cuanto mayor sea el número de hijos, mayor será la probabilidad de abstracción incorrecta de la clase padre. Si una clase tiene una gran cantidad de niños, puede ser un caso de mal uso de subclases.
- El número de niños da una idea de la influencia potencial que tiene una clase en el diseño. Si una clase tiene una gran cantidad de niños, puede requerir más pruebas de los métodos en esa clase.

En resumen, la métrica NOC puede ser útil para identificar clases que pueden estar sobrecargadas con demasiadas responsabilidades. Un NOC alto puede indicar problemas de diseño y violaciones del principio de responsabilidad única.

CBO (coupling between objects o acoplamiento entre objetos)

Es una medida que indica la cantidad de clases a las que una clase está acoplada a través de referencias a otras clases. El acoplamiento se refiere a la interdependencia entre dos objetos, y se produce cuando los métodos y/o variables de instancia de una clase son utilizados por otra. En otras palabras, dos clases están acopladas si los métodos de una clase hacen uso de los métodos y/o variables de instancia de la otra.

Puntos de vista:

- El acoplamiento excesivo entre clases de objetos es perjudicial para el diseño modular y evita la reutilización. Cuanto más independiente sea una clase, más fácil será reutilizarla en otra aplicación.
- Para mejorar la modularidad y promover la encapsulación, los pares entre clases de objetos deben mantenerse al mínimo. Cuanto mayor sea el número de parejas, mayor será la sensibilidad a los cambios en otras partes del diseño y, por lo tanto, el mantenimiento será más difícil.
- Una medida de acoplamiento es útil para determinar cuán complejas pueden ser las pruebas de varias partes de un diseño. Cuanto mayor sea el acoplamiento de clase entre objetos, más rigurosa debe ser la prueba.

Un alto valor en esta métrica indica una dependencia fuerte y una baja capacidad de reutilización del código. En términos generales, una alta dependencia indica que una clase está altamente acoplada con otras clases, lo que hace que sea más difícil modificarla sin afectar a otras clases en el sistema.

RFC (response for a class o respuesta para una clase)

Mide la cantidad de métodos en una clase que pueden ser llamados en respuesta a una solicitud de un objeto externo.

Puntos de vista:

- Si se puede invocar una gran cantidad de métodos en respuesta a un mensaje, la prueba y depuración de la clase se vuelve más complicada ya que requiere un mayor nivel de comprensión por parte del probador.
- Cuanto mayor sea el número de métodos que se pueden invocar desde una clase, mayor será la complejidad de la clase.
- Tener en cuenta el valor máximo de posibles respuestas puede ser útil para asignar el tiempo de prueba necesario de manera adecuada y exhaustiva.
- Un número pequeño de clases, generará usualmente un alto número de métodos.

Un alto valor de RFC indica una complejidad potencialmente alta de la clase, lo que puede dificultar la comprensión y el mantenimiento del código. Por lo tanto, es recomendable mantener una RFC baja para facilitar la evolución y el mantenimiento del software.

LCOM (lack of cohesion in methods o falta de cohesión en métodos)

Se utiliza para medir la cohesión de una clase, es decir, la medida en que los métodos de una clase están relacionados entre sí. LCOM se basa en el principio de que una clase debe estar diseñada de manera que los métodos que pertenecen a ella estén estrechamente relacionados, trabajen juntos y compartan información de manera significativa. Una clase con alta cohesión es más fácil de entender, mantener y extender.

Puntos de vista:

- Es deseable la cohesión de los métodos dentro de una clase, ya que promueve la encapsulación.
- La falta de cohesión implica que las clases probablemente deberían dividirse en dos o más subclases.
- Cualquier medida de disparidad de métodos ayuda a identificar fallas en el diseño de clases. La baja cohesión aumenta la complejidad, lo que aumenta la probabilidad de errores durante el proceso de desarrollo.

9) Adecuación funcional (AF)

El término “adecuación funcional” ha sido utilizado en diferentes contextos, y su definición ha evolucionado con el tiempo. Según la IEEE Standard Glossary of Software Engineering Terminology [52], la AF se define como “la capacidad de un software para proporcionar funciones que satisfagan las necesidades establecidas o implícitas”. En el contexto de la IS, la AF es fundamental para garantizar que un programa cumpla con los requisitos del usuario. Un software adecuado funcionalmente debe proporcionar todas las características y funcionalidades necesarias para satisfacer las necesidades del usuario, sin errores o fallos que puedan afectar su desempeño [112].

En la actualidad, los sistemas de software se han convertido en una parte esencial en la vida diaria. Sin embargo, en muchas ocasiones, estos sistemas no funcionan como se espera, lo que hace que crear software de alta calidad sea un desafío intelectual. Para garantizar la calidad del software, se lleva a cabo una actividad de prueba, aunque esta requiere mucho tiempo y recursos. Es esencial asegurar un cierto porcentaje de calidad del software, ya que un defecto en el mismo puede tener graves consecuencias para los usuarios y las empresas [147]. Estas consecuencias pueden ser desde problemas triviales, como pérdida de dinero y tiempo, hasta la pérdida de credibilidad comercial o incluso la pérdida de vidas. Por lo tanto, la AF es uno de los atributos más importantes para determinar la calidad del software [54].

a) Evaluación de la AF

La evaluación de la AF es un proceso importante en el desarrollo de software, que implica la realización de pruebas para asegurarse de que el software funciona como se espera y cumple con los requisitos del usuario [2]. La evaluación de la AF se realiza en diferentes etapas del ciclo de vida del software. En la etapa de diseño, se evalúa la AF del software para asegurarse de que se han incluido todas las características necesarias para satisfacer los requisitos del usuario. Durante la fase de desarrollo, se realizan pruebas para evaluar la AF del software en diferentes escenarios de uso y con diferentes configuraciones de hardware y software. En la etapa de implementación, se realizan pruebas finales para asegurarse de que el software cumple con los requisitos del usuario.

Para realizar una evaluación de la AF, se deben identificar los requisitos del usuario y definir los criterios de aceptación. Estos criterios de aceptación se utilizan para evaluar la funcionalidad del software y asegurarse de que cumple con los requisitos del usuario. Los criterios de aceptación pueden incluir pruebas de rendimiento, pruebas de usabilidad, pruebas de compatibilidad, pruebas de seguridad, entre otras.

Según la ISO/IEC 25010 [148], la AF se divide en las siguientes subcaracterísticas:

Completitud funcional. Grado en el cual el conjunto de funcionalidades cubre todas las tareas y los objetivos del usuario especificados.

Corrección funcional. Capacidad del producto o sistema para proveer resultados correctos con el nivel de precisión requerido.

Pertinencia funcional. Capacidad del producto software para proporcionar un conjunto apropiado de funciones para tareas y objetivos de usuario especificados.

Estas subcaracterísticas pueden ser utilizadas como guía para definir métricas específicas que midan cada aspecto de la calidad del software relacionado con su AF. Por ejemplo, se puede definir una métrica para medir la completitud funcional del software, o una métrica para medir la exactitud de los resultados producidos por el software, utilizando un método de lista de verificación que contiene funciones que han sido diseñadas con anterioridad y luego serán validadas por uno o varios evaluadores expertos en el desarrollo de sistemas de información [54], [149]–[151].

Por otro lado, los casos de prueba son una técnica utilizada en el proceso de pruebas de software para evaluar si el software cumple con los requisitos funcionales establecidos [2]. Consiste en diseñar una serie de situaciones que prueben la funcionalidad del software y que permitan identificar posibles errores o problemas de funcionamiento [152]. Además, son importantes porque permiten verificar que el software se comporta como se espera y que cumple con las especificaciones definidas. Además, ayudan a identificar los errores en el software antes de que sea entregado al cliente, lo que puede ahorrar tiempo y recursos en correcciones posteriores

Una de las técnicas más usadas y fiables para evaluar la AF del software es la técnica de caja negra [2]. En esta técnica, el evaluador no tiene conocimiento del código fuente del software, sino que se enfoca en probar el software desde el punto de vista del usuario final. Se prueban diferentes escenarios y entradas, y se verifica si el software produce las salidas correctas [30]. En general, los casos de prueba son una técnica importante y ampliamente utilizada para evaluar la AF del software, y su diseño y ejecución adecuados pueden ayudar a garantizar un alto nivel de calidad en el software.

C. FORMULACIÓN DE HIPOTESIS

En este estudio, se busca determinar si la formación de grupos homogéneos basados en la dimensión de la meticulosidad se traduce en mejores resultados en términos de la mantenibilidad y la adecuación funcional del producto software en comparación con la formación de grupos basados en las preferencias de los estudiantes. Con base en esto, se plantean las siguientes hipótesis:

Con respecto a la mantenibilidad, se formuló las siguientes hipótesis:

$H_{1,0}$ = GHBDM no logra mejores resultados con respecto a la mantenibilidad frente a GFPE.

$H_{1,1}$ = GHBDM logra mejores resultados con respecto a la mantenibilidad frente a GFPE.

Con respecto a la adecuación funcional, se formuló las siguientes hipótesis:

$H_{2,0}$ = GHBDM no logra mejores resultados con respecto a la adecuación funcional frente a GFPE.

$H_{2,1}$= GHBDM logra mejores resultados con respecto a la adecuación funcional frente a GFPE.

III. METODOLOGÍA

A. TIPO DE INVESTIGACIÓN

El proceso investigativo se desarrolló bajo el paradigma positivista, ya que se fundamenta en el conocimiento científico, con un enfoque cuantitativo el cual permite la recolección y el análisis de datos numéricos, fundamental para la medición y comparación de variables. Además, la investigación cuantitativa permite un mayor control sobre las variables y una mayor precisión en la medición de los resultados. En este estudio, se midieron los rasgos de personalidad de los estudiantes, las variables para evaluar la mantenibilidad y la adecuación funcional del software desarrollado por los grupos utilizando métodos cuantitativos. Además, se aplicaron técnicas estadísticas para analizar los datos obtenidos y establecer relaciones entre las variables de interés. En este sentido, se empleó un tipo de investigación correlacional que permitió medir el grado de relación entre las variables de estudio y se utilizó un diseño experimental general basado en un cuasi-experimento, tal como se muestra en la TABLA III.

TABLA III
DISEÑO EXPERIMENTAL

	Estímulo	Post-prueba
G_1 Grupo experimental	X	O_1
G_2 Grupo de control	—	O_2

Fuente: esta investigación.

La validación de este estudio, se realizó en grupos diferentes pertenecientes a cursos de ingeniería de software III del Programa de Ingeniería de Sistemas de la Universidad de Nariño (sede Pasto e Ipiales), Colombia. Para el experimento se consideraron dos grupos, experimental (G_1) y de control (G_2). Donde X fue el tratamiento experimental, el cual consiste en formar grupos homogéneos basados en la dimensión de la meticulosidad. El grupo de control son los grupos formados por preferencia de los estudiantes. Al finalizar el experimento, se aplicaron post-pruebas (O_1 y O_2) para evaluar la mantenibilidad y la adecuación funcional del software desarrollado por ambos grupos.

B. DISEÑO DE LA INVESTIGACIÓN

El propósito de esta investigación fue determinar si la formación de grupos homogéneos basados en la dimensión de la meticulosidad genera mejores resultados en términos de mantenibilidad y AF del producto de software en comparación con los grupos formados por preferencia de los estudiantes.

Para llevar a cabo este estudio, se propuso una metodología que consta de cinco etapas secuenciales, las cuales se detallan en las siguientes secciones. Las cajas situadas en la parte superior representan las entradas requeridas para cada etapa, mientras que las cajas inferiores describen las salidas obtenidas en cada una de ellas. Es importante aclarar que este proceso es realizado para entornos académicos. La Fig. 3 proporciona una síntesis gráfica de este proceso.

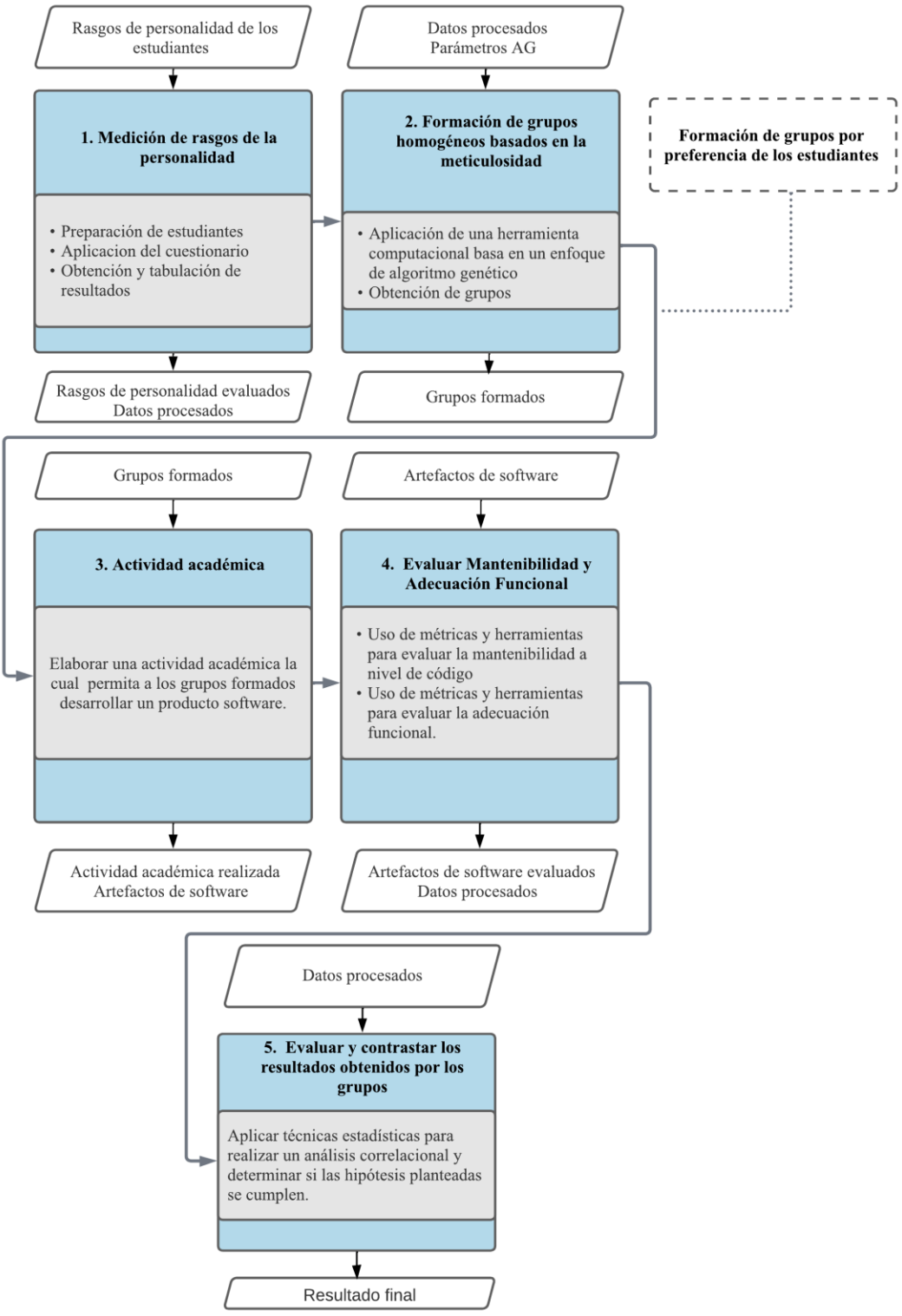


Fig. 3. Esquema metodológico.

Fuente: esta investigación.

1) Medición de rasgos de la personalidad

La evaluación de rasgos de personalidad en el campo de la IS es una tarea crucial, y para ello se utilizan diversos modelos e instrumentos psicológicos [20]. Sin embargo, la gran cantidad de opciones disponibles puede generar diferencias en los resultados obtenidos, lo que puede afectar la validez y confiabilidad de los mismos [20], [21], [153], [154]. Por esta razón, es importante realizar una selección cuidadosa y rigurosa de los modelos e instrumentos a utilizar, y asegurarse de que sean validados y confiables en la población en la que se aplicarán. De esta manera, se garantiza que los resultados obtenidos sean lo más precisos y confiables posible, lo que permitirá realizar una evaluación más acertada para la formación de grupos homogéneos basados en la meticulosidad.

Se aclara que, para efectos de este estudio se tuvo en cuenta el proceso puramente cuantitativo, no se consideraron aspectos cualitativos relacionados con la personalidad de los estudiantes participantes, por estar fuera del alcance del estudio.

Con el objetivo de analizar y extraer de forma confiable y válida los rasgos de personalidad de los desarrolladores de software, se llevó a cabo un estudio de mapeo sistemático (SMS, por sus siglas en inglés). Este mapeo permitió identificar y sintetizar la evidencia disponible sobre la evaluación de los rasgos de personalidad en este campo y, a su vez, ayudó a seleccionar el modelo e instrumento psicológico más adecuado para su aplicación. Asimismo, el mapeo contribuye a identificar brechas en la investigación actual y a proporcionar recomendaciones para futuras investigaciones en esta área.

Es importante destacar que llevar a cabo un SMS no es una tarea sencilla. Por consiguiente, con el fin de asegurar la calidad del estudio y la validez de los resultados obtenidos, se siguieron los pasos establecidos por Petersen et al. [155]. Estos pasos se muestran en la Fig. 4. Con este enfoque riguroso y sistemático, se obtuvo una visión completa y confiable de la evidencia disponible sobre la evaluación de los rasgos de personalidad en la IS.

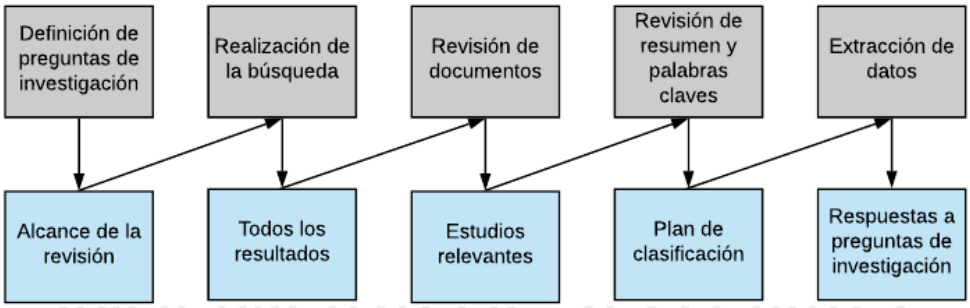


Fig. 4. Proceso de mapeo sistemático.

Fuente: Tomado de Petersen et al. [155].

A continuación, se describe brevemente cada fase para realizar el SMS [155]:

- **Definición de la pregunta de investigación:** Se debe definir claramente la pregunta de investigación que se quiere responder con el estudio de mapeo sistemático. Esto ayudará a establecer los criterios de inclusión y exclusión y a seleccionar las fuentes de información relevantes.

- **Realización de la búsqueda:** Se debe realizar una búsqueda sistemática y exhaustiva de fuentes de información relevantes. Para identificar los estudios primarios, se emplean cadenas de búsqueda en bases de datos científicas, o navegando manualmente a través de actas de congresos o publicaciones de revistas relevantes. Para esta tarea se debe utilizar bases de datos bibliográficas válidas y confiables como ACM Digital Library, IEEE Digital Library, Scopus, entre otras.
- **Revisión de documentos:** Una vez realizada la búsqueda, se debe realizar una revisión profunda de los documentos identificados para determinar si cumplen con los criterios de inclusión y exclusión previamente establecidos.
- **Revisión de resúmenes y palabras claves:** En este paso, se procederá a la revisión de los resúmenes de los artículos previamente seleccionados para identificar aquellos que sean relevantes y adecuados a la pregunta de investigación. Se buscarán palabras clave y conceptos que reflejen su contribución y se evaluará su calidad en función de la claridad y precisión de la información proporcionada. En caso de que los resúmenes no sean lo suficientemente claros para permitir la elección de palabras clave significativas, se examinarán también las secciones de introducción o conclusión del artículo. Una vez que se ha seleccionado un conjunto final de palabras clave, se agruparán y utilizarán para formar las categorías del mapa.
- **Extracción de datos y respuesta a las preguntas de investigación:** Finalmente, se extraen y registran los datos relevantes de cada artículo incluido en el estudio y se analizan para responder a las preguntas de investigación planteadas. Este paso es esencial para obtener conclusiones significativas y relevantes a partir del SMS. Con este enfoque, se espera obtener una visión clara y completa del panorama actual de la evaluación de los rasgos de personalidad en la IS y facilitar la toma de decisiones informadas en futuros proyectos de investigación.

a) Preparación de estudiantes

La preparación adecuada de los estudiantes antes de aplicar el cuestionario para evaluar sus rasgos de personalidad, es crucial para garantizar que el proceso sea lo más preciso, auténtico y respetuoso posible. Esta evaluación puede ser un proceso delicado y emocionalmente intenso. Los estudiantes pueden sentirse incómodos o ansiosos al responder preguntas personales y potencialmente intrusivas. Por lo tanto, el docente a cargo brindó una explicación clara y detallada sobre el propósito y la finalidad del proceso que se llevará a cabo. En el caso específico del cuestionario, se recomendó los estudiantes que lo completen con sinceridad y de manera exhaustiva, ya que no existen respuestas "correctas" o "incorrectas". Es importante aclarar que los resultados proporcionan un perfil general de la personalidad de los participantes, los cuales fueron utilizados únicamente para la conformación de grupos homogéneos basados en la meticulosidad. En ningún momento se pretendió emitir algún tipo de concepto o diagnóstico psicológico de los participantes en el estudio, pues esto se encuentra fuera del alcance del mismo.

Además, los expertos sugieren que se registre la aceptación de los estudiantes mediante la cumplimentación de un "Consentimiento Informado", en el cual se establece el uso específico de los resultados, en este caso, la conformación de grupos con fines educativos. Este proceso puede realizarse en un formato físico o digital. Se adjunta una muestra del formato utilizado en el Anexo *1* del presente estudio.

b) El cuestionario

Para una explicación detallada del cuestionario utilizado en este estudio, es necesario considerar los resultados obtenidos en el SMS mencionado anteriormente. En este estudio se encontró que, en los últimos años, los instrumentos psicométricos basados en el modelo Big Five, como el BFI (Big Five Inventory), han cobrado mayor relevancia en la evaluación de la personalidad de los desarrolladores de software en el campo de la IS [48]. BFI consta de 44 ítems de respuesta múltiple (tipo Likert) que miden las dimensiones propuestas por el modelo Big Five: Extroversión, Amabilidad, Meticulosidad, Neuroticismo y Apertura a Nuevas Experiencias. En la sección de resultados de este estudio se proporcionará información de este SMS.

La Fig. 5 presenta la versión en español del instrumento BFI desarrollado por John et al. [83], denominado "BFI español", el cual fue obtenido del estudio titulado "Los Cinco Grandes Across Cultures and Ethnic Groups: Multitrait Multimethod Analyses of the Big Five in Spanish and English" [156]. En la TABLA IV se muestran estos ítems organizados por cada una de las dimensiones.

Las siguientes expresiones le describen a usted con más o menos precisión. Por ejemplo, ¿está de acuerdo en que usted es alguien "*chistoso, a quien le gusta bromear*"? Por favor escoja un número para cada una de las siguientes expresiones, indicando así hasta qué punto está de acuerdo o en desacuerdo en cómo le describe a usted.

Muy en desacuerdo	Ligeramente en desacuerdo	Ni de acuerdo ni en desacuerdo	Ligeramente de acuerdo	Muy de acuerdo
1	2	3	4	5

Me veo a mi mismo-a como *alguien que…*

___ 1. es bien hablador
___ 2. tiende a ser criticón
___ 3. es minucioso en el trabajo
___ 4. es depresivo, melancólico
___ 5. es original, se le ocurren ideas nuevas
___ 6. es reservado
___ 7. es generoso y ayuda a los demás
___ 8. puede a veces ser algo descuidado
___ 9. es calmado, controla bien el estrés
___ 10. tiene intereses muy diversos
___ 11. está lleno de energía
___ 12. prefiere trabajos que son rutinarios
___ 13. inicia disputas con los demás
___ 14. es un trabajador cumplidor, digno de confianza
___ 15. con frecuencia se pone tenso
___ 16. tiende a ser callado
___ 17. valora lo artístico, lo estético
___ 18. tiende a ser desorganizado
___ 19. es emocionalmente estable, difícil de alterar
___ 20. tiene una imaginación activa
___ 21. persevera hasta terminar el trabajo
___ 22. es a veces mal educado con los demás
___ 23. es inventivo
___ 24. es generalmente confiado
___ 25. tiende a ser flojo, vago
___ 26. se preocupa mucho por las cosas
___ 27. es a veces tímido, inhibido
___ 28. es indulgente, no le cuesta perdonar
___ 29. hace las cosas de manera eficiente
___ 30. es temperamental, de humor cambiante
___ 31. es ingenioso, analítico
___ 32. irradia entusiasmo
___ 33. es a veces frío y distante
___ 34. Hace planes y los sigue cuidadosamente
___ 35. mantiene la calma en situaciones difíciles
___ 36. le gusta reflexionar, jugar con las ideas
___ 37. es considerado y amable con casi todo el mundo
___ 38. se pone nervioso con facilidad
___ 39. es educado en arte, música, o literatura
___ 40. es asertivo, no teme expresar lo que quiere
___ 41. le gusta cooperar con los demás
___ 42. se distrae con facilidad
___ 43. es extrovertido, sociable
___ 44. tiene pocos intereses artísticos

Por favor, compruebe que ha escrito un número delante de cada frase.

Note. Copyright 1991 by Oliver P. John.

Fig. 5. BFI versión español.

Fuente: Tomado de John et al. [83].

TABLA IV
ÍTEMS DEL BFI ORGANIZADOS POR DIMENSIÓN

Dimensión	Ítem	Descripción
Extraversion (Extraversión)	1	Es bien hablador.
	6	Es reservado.
	11	Está lleno de energía.
	16	Tiende a ser callado.
	27	Es a veces tímido, inhibido.
	32	Irradia entusiasmo.
	40	Es asertivo, no teme expresar lo que quiere.
	43	Es extrovertido, sociable.
Agreeableness (Amabilidad)	2	Tiende a ser criticón.
	7	Es generoso y ayuda a los demás.
	13	Inicia disputas con los demás.
	22	Es a veces maleducado con los demás.
	24	Es generalmente confiado.
	28	Es indulgente, no le cuesta perdonar.
	33	Es a veces frío y distante.
	37	Es considerado y amble con casi todo el mundo.
	41	Le gusta cooperar con los demás.
Conscientiousness (Meticulosidad)	3	Es minucioso en el trabajo.
	8	Puede ser a veces algo descuidado.
	14	Es un trabajador, digno de confianza.
	18	Tiende a ser desorganizado.
	21	Persevera hasta terminar el trabajo.
	25	Tiende a ser flojo, vago.
	29	Hace las cosas de manera eficiente.
	34	Hace planes y los sigue cuidadosamente.
	42	Se distrae con facilidad.
Neuroticism (Inestabilidad Emocional)	4	Es depresivo, melancólico.
	9	Es clamado, controla bien el estrés.
	15	Con frecuencia se pone tenso.
	19	Es emocionalmente estable, difícil de alterar.
	26	Se preocupa mucho por las cosas.
	30	Es temperamental, de humor cambiante.
	35	Mantiene la calma en situaciones difíciles.
	38	Se pone nervioso con facilidad.
Openness (Apertura a Nuevas Experiencias)	5	Es original, se le ocurren ideas nuevas.
	10	Tiene intereses muy diversos.
	12	Prefiere trabajos que son rutinarios.
	17	Valora lo artístico, lo estético.
	20	Tiene una imaginación activa.
	23	Es inventivo.
	31	Es ingenioso, analítico.
	36	Le gusta reflexionar, jugar con las ideas.
	39	Es educado en arte, música o literatura.
	44	Tiene pocos intereses artísticos.

Fuente: Tomando de John et al. [83].

c) ***Aplicación del cuestionario***

Una vez que los estudiantes han sido debidamente preparados, se procede a la aplicación del cuestionario. Esta aplicación puede llevarse a cabo en formato impreso o de manera digital, ya sea mediante un formulario en PDF, una hoja electrónica o aplicaciones informáticas diseñadas específicamente para este propósito. En este estudio, el cuestionario se subió en la plataforma de

Moodle de la institución universitaria para automatizar todo el proceso, desde la aplicación de los cuestionarios hasta la obtención de los resultados correspondientes. En el Anexo 2 se incluye una captura de pantalla de la aplicación con su URL correspondiente, la cual estará disponible de manera gratuita para fines académicos o de investigación previa comunicación con los desarrolladores.

d) Obtención y tabulación de resultados

Una vez se ha aplicado el cuestionario, se obtuvieron las puntuaciones individuales de los estudiantes para cada dimensión. La obtención de los resultados puede realizarse de forma manual o mediante el uso de una herramienta computacional automatizada. En ambos casos, se sugiere presentar una tabla de resultados, como se muestra en la TABLA V:

TABLA V
TABULACIÓN SUGERIDA DE RESULTADOS

ID	Nombre	E	A	M	N	AP
1	Estudiante A	4.50	4.00	3.5	2.25	3
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.

Fuente: esta investigación.

Cada letra representa las dimensiones del modelo Big Five: E (extroversión), A (amabilidad), M (meticulosidad), N (neuroticismo) y AP (apertura a nuevas experiencias).

e) Exportación de datos

Después de tabular los resultados, se debe generar un archivo de texto plano que contenga los valores separados por comas, el cual es utilizado como entrada para el algoritmo de agrupamiento. Es importante considerar las posibles situaciones que pueden surgir durante este proceso, como la presencia de estudiantes que ya no forman parte del curso al momento de la conformación de los grupos, ya sea porque se retiraron o por otra razón. En estos casos, es necesario eliminar dichos registros del archivo para evitar que interfieran con el proceso de agrupamiento.

En la Fig. 6 se presenta una muestra de cómo se prepararía el mencionado archivo.

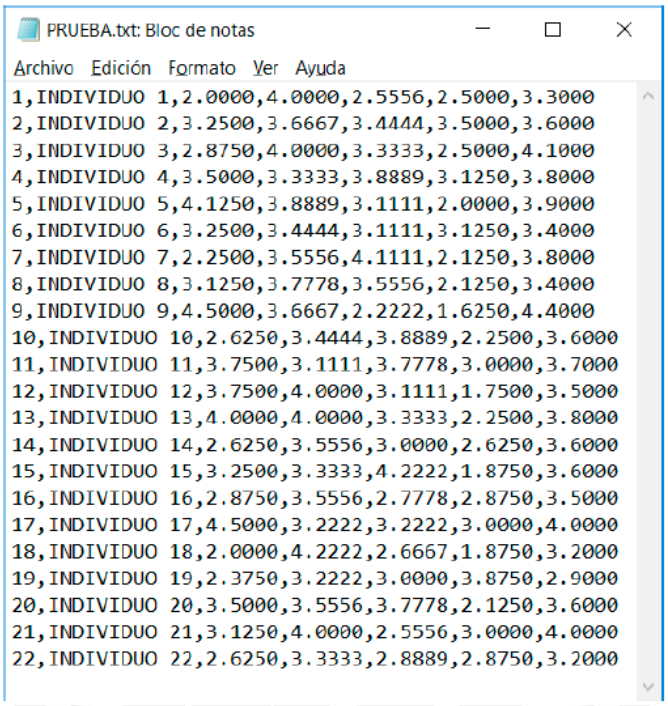


Fig. 6. Muestra del archivo de texto plano.

Fuente: esta investigación.

Como puede observarse, cada campo del archivo se encuentra separado por una coma, siguiendo la sugerencia de tabulación descrita anteriormente. El orden de los campos es el siguiente: identificación, nombre (se ha omitido el nombre real para preservar la confidencialidad de la información de los participantes), extraversión, amabilidad, meticulosidad, neuroticismo y apertura a nuevas experiencias.

2) Formación de grupos homogéneos basados en la meticulosidad

La conformación de grupos homogéneos se refiere a la organización de individuos con características similares dentro de un conjunto más amplio [157]. Este proceso puede ser complejo, ya que es necesario definir criterios claros y objetivos para identificar las similitudes entre los miembros del grupo [158]. Además, puede haber diversas variables que influyan en la homogeneidad del grupo, como la edad, género, nivel socioeconómico, educación, intereses, entre otros. Por lo tanto, la creación de grupos homogéneos puede requerir un análisis cuidadoso y un enfoque personalizado para garantizar la efectividad y el éxito del grupo [157].

Para el presente estudio, se utilizó la herramienta computacional propuesta por Sánchez et al. [28] para formar grupos homogéneos basados en la dimensión de la meticulosidad. Dicha herramienta utiliza un enfoque de algoritmo genético (AG) que se basa en los rasgos de personalidad de los estudiantes como criterio de agrupación. Dicha herramienta tiene como objetivo mejorar la calidad de la formación de grupos homogéneos de estudiantes basados en rasgos de personalidad.

En la Fig. 7 se muestra el flujo del algoritmo genético interno para este fin.

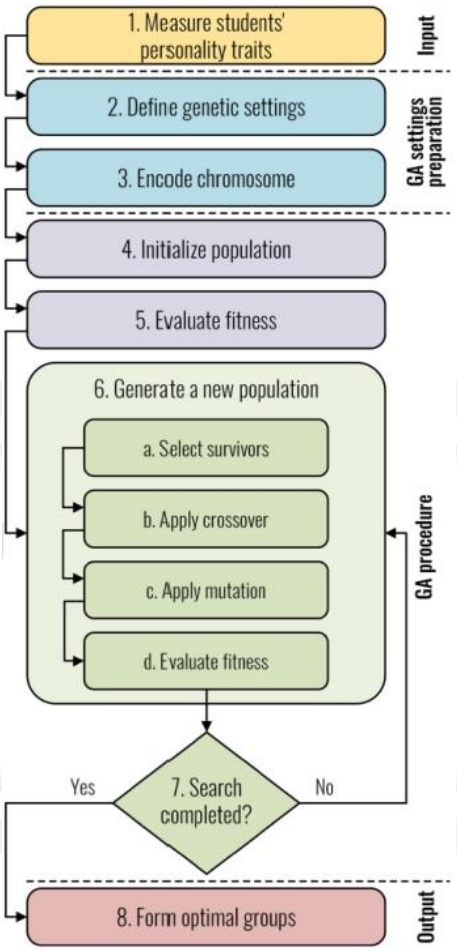


Fig. 7. Flujo del proceso de formación de grupos homogéneos mediante AG.

Fuente:Tomado de Sánchez et al. [28].

A continuación, se describen brevemente los pasos involucrados en el proceso de formación de grupos [28]:

Paso 1 - Medición de los rasgos de personalidad de los estudiantes: El primer paso es medir las características de los estudiantes en base a las cuales se forman los grupos, en este caso, sus rasgos de personalidad. Esto es crucial para estructurar buenos grupos que fomenten una colaboración eficiente y efectiva y logren mejores resultados en sus actividades.

Paso 2 - Definir parámetros genéticos: Antes de ejecutar el algoritmo genético, se deben establecer los parámetros genéticos relativos al tamaño del grupo, tamaño de la población, número de generaciones y probabilidades de cruzamiento y mutación.

Paso 3 - Codificar cromosoma: En este paso, se representa el cromosoma (que contiene información sobre los estudiantes) en una estructura de datos predefinida para permitir la aplicación de los operadores genéticos.

Paso 4 - Inicializar población: El algoritmo genético comienza creando una población inicial que consta de ciertas soluciones codificadas viables (cromosomas). Esta población se genera al azar para asegurar su diversidad.

Paso 5 - Evaluar la aptitud: Se utiliza una función de aptitud basada en los rasgos de personalidad de los estudiantes para evaluar los cromosomas de la población. Un valor de aptitud más bajo para un cromosoma representa una mejor solución.

Paso 6 - Generar una nueva población: Este proceso es el corazón del algoritmo genético, donde se generan soluciones nuevas y mejores. Los operadores genéticos aplicados en este paso son los siguientes:

- Selección, donde se seleccionan los individuos a clonar (sobrevivientes);
- Cruzamiento, donde, basado en una probabilidad, se realiza una recombinación de los genes de los padres para producir una descendencia mejor;
- Mutación, donde, basado en una probabilidad, se mutan partes del cromosoma de la nueva población;
- Se evalúa la aptitud de los cromosomas de la población.

Paso 7 - Fin de la búsqueda: Después de varias generaciones, el algoritmo termina y converge al cromosoma más apto, que representa la solución óptima.

Paso 8 - Formación de grupos óptimos: Se forman grupos de estudiantes en base a los resultados del algoritmo genético, y se notifica a los estudiantes para comenzar a trabajar en sus grupos en el desarrollo de la actividad académica propuesta.

a) Aplicación de la herramienta computacional para formar los grupos homogéneos

La herramienta computacional está desarrollada en Java, su biblioteca “Team-B v2.2” permite la creación de aplicaciones Java de escritorio, web o móviles que permitan la formación automática y homogénea de grupos en escenarios de aprendizaje colaborativo, incluso con individuos que presenten diferentes características cuantificables. La interfaz de usuario presentada en la Fig. 8 permite la formación de grupos homogéneos de manera sencilla y eficiente.

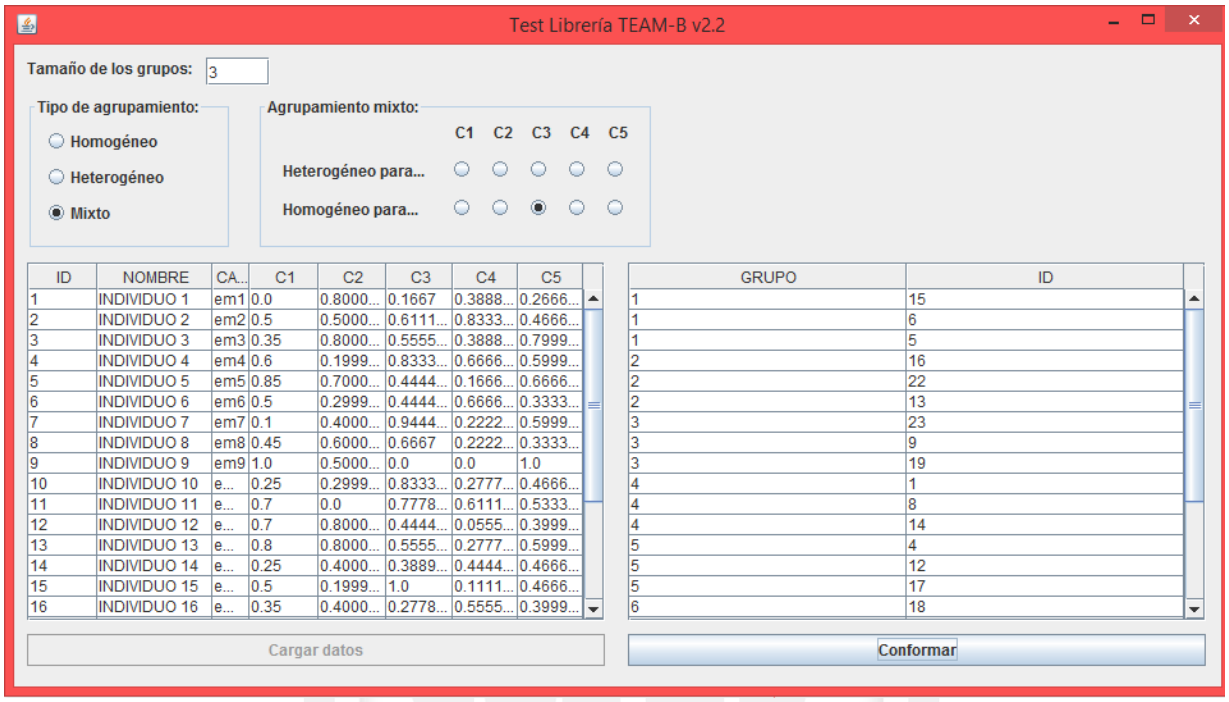


Fig. 8. Interfaz de la herramienta para formar los grupos homogéneos basados en la meticulosidad.

Fuente: esta investigación.

Como se mencionó anteriormente, es importante evaluar los rasgos de personalidad de los estudiantes como parámetro de entrada para utilizar esta herramienta. Para ello, se requiere la importación de un archivo de texto plano con valores separados por comas y con una estructura similar a la que se muestra en la Fig. 6. Es importante tener en cuenta la posibilidad de que el número total de estudiantes no sea múltiplo del tamaño de los grupos necesarios. Si esto sucede, se recomienda complementar con "dummies" los estudiantes faltantes para crear grupos completos. Se les debe asignar como valor en cada una de las características consideradas la media del grupo total en cada una de ellas. De esta manera, se puede garantizar la equidad en la distribución de los estudiantes en los grupos y evitar posibles desigualdades en el proceso de asignación.

b) Obtención de grupos

La Fig. 8 muestra la interfaz de usuario que se utilizó para formar los grupos homogéneos basados en la meticulosidad. El primer parámetro es el "Tamaño de los grupos", el cual en el ejemplo de prueba está establecido en 3. El segundo parámetro es el "Tipo de agrupamiento" y está establecido en "Mixto". El tercer parámetro es el "Agrupamiento mixto", que está establecido como "homogéneo para C3", donde C3 corresponde a la dimensión de la meticulosidad. Además, se visualiza una tabla que contiene la información de los estudiantes que se cargaron desde el texto plano, incluyendo sus características, las cuales se explicaron anteriormente. En la parte derecha de la interfaz, se muestra una tabla con los grupos homogéneos formados. Por ejemplo, el Grupo 1 estaría compuesto por los estudiantes con ID 15, 6 y 5, y así sucesivamente con los demás grupos. Es importante destacar que este procedimiento se llevó a cabo específicamente para el grupo experimental. En cuanto al grupo de control, se formaron por preferencia de los estudiantes.

3) Actividad académica

Las actividades académicas son fundamentales para el desarrollo de habilidades y conocimientos en los estudiantes y también para el avance del conocimiento en la comunidad académica. Las actividades académicas pueden ser individuales o en grupo, y pueden tener diversos objetivos, como el desarrollo de habilidades específicas, la aplicación de conceptos teóricos en situaciones prácticas, la investigación y el análisis crítico de temas relevantes, entre otros [159].

En el caso específico del desarrollo de un producto software en una actividad académica, es importante para los estudiantes ya que les permite aplicar y poner en práctica los conocimientos teóricos y habilidades técnicas que han adquirido en el aula. Además, les proporciona una experiencia realista del proceso de desarrollo de software, lo que les permite familiarizarse con las metodologías y herramientas utilizadas en la industria. También les brinda la oportunidad de trabajar en grupo y desarrollar habilidades de colaboración, comunicación y liderazgo, entre otras habilidades blandas valiosas para su formación integral [160].

Elaborar una actividad académica la cual permita a los grupos formados desarrollar un producto software.

Objetivo: Fomentar el aprendizaje colaborativo, el desarrollo de habilidades técnicas y de comunicación, y la aplicación de los conocimientos adquiridos en la creación de un producto de software, teniendo en cuenta la formación de grupos.

Duración: 6 semanas

Participantes: Estudiantes de quinto semestre del Programa de Ingeniería de Sistemas de la Universidad de Nariño (sede Pasto e Ipiales), Colombia. Estos alumnos cursaban la asignatura de Ingeniería de Software III.

En la TABLA VI**¡Error! No se encuentra el origen de la referencia.** se describen los pasos de la actividad académica:

TABLA VI
PASOS DE LA ACTIVIDAD ACADÉMICA

Pasos	Descripción
Presentar la actividad académica	El docente a cargo de la actividad académica presenta y explica los objetivos de la misma.
Definir los requisitos	Con el fin de que los estudiantes se concentren únicamente en la implementación del software, se crearon previamente los requisitos del producto. Se tomó en cuenta que el software a desarrollar no estuviera disponible en Internet para evitar que se reutilice código ya existente. De esta manera, se asegura que los estudiantes se comprometan a crear un producto original. El Anexo 3 incluye la descripción del problema a desarrollar, acompañado de las respectivas historias de usuario (HU). De esta manera, se proporciona a los estudiantes toda la información necesaria para comenzar con la implementación del software, permitiéndoles entender claramente las necesidades y expectativas del usuario final.

TABLA VI
PASOS DE LA ACTIVIDAD ACADÉMICA

Pasos	Descripción
Diseñar y planificar	Los grupos diseñan y planifican el desarrollo del software, estableciendo las tareas a realizar y su duración, asignando responsabilidades a cada miembro del grupo.
Desarrollar el producto software	Los grupos trabajan en la creación del software, siguiendo el plan establecido. Durante esta etapa, es importante que realicen reuniones periódicas para revisar el avance del proyecto y realizar ajustes si fuera necesario. De esta manera, podrán asegurarse de cumplir con los plazos establecidos y que el software se esté desarrollando de acuerdo a las expectativas del usuario final.
Supervisar el proceso	El proceso de entrega del software es incremental, con revisiones semanales por parte del docente para guiar y revisar el progreso de los grupos.
Presentación del software	Los grupos deben presentar el software desarrollado ante el docente y los demás compañeros de clase, explicando el proceso de desarrollo, las dificultades encontradas y las soluciones aplicadas. El docente evalúa la AF del producto software utilizando los casos de prueba, cuyo diseño se presenta en el Anexo 4. El docente evalúa el trabajo de los grupos, tomando en cuenta la calidad del software desarrollado, la capacidad de trabajo en equipo, el cumplimiento de los requisitos establecidos, y la calidad de la presentación realizada.
Entrega del software	Los grupos deben entregar el software terminado en la fecha establecida por el docente. Además, deberán entregar toda la documentación necesaria, como código fuente, base de datos, guía de instalación, para facilitar posteriormente la evaluación del producto software.

Fuente: esta investigación.

Para garantizar la relevancia del producto software, se planteó un problema cercano a la realidad que permita a los estudiantes demostrar su conocimiento sobre la metodología ágil de desarrollo (SCRUM) parte del curso de ingeniería de software III, programación orientada a objetos y base de datos, contemplados y aprobados en anteriores cursos, así como su capacidad para analizar y desarrollar soluciones de software para el problema planteado.

Se les informó a los grupos que el producto de software solo se evaluará en función de dos atributos de calidad: la mantenibilidad y la AF. Por lo tanto, se les pidió que no hicieran énfasis y esfuerzos incensarios en otros aspectos como el diseño, seguridad, eficiencia, etc. Con esto se aseguró que el experimento se centre en los atributos elegidos y se evite distraer a los estudiantes con aspectos que no serán evaluados. Esto permitió una evaluación más objetiva y centrada en los objetivos del experimento. De igual forma, se les comunicó que todos miembros de los grupos deben tener el rol de desarrolladores, donde participen en todo el proceso de desarrollo y adquieran una comprensión completa de cómo se construye un producto de software. Por último, se les indicó que podrían resolver cualquier duda sobre los requisitos del usuario y/o historias de usuario a través de Microsoft Teams [161]. Para ello se crearon dos grupos en dicha plataforma, uno para la sede de Ipiales y otro para la sede de Pasto.

4) Evaluación de la mantenibilidad y adecuación funcional del producto software

Para el presente estudio, se decidió que la evaluación de la mantenibilidad del producto software estará orientada en el paradigma OO. Dicho enfoque se debe a que el producto software a desarrollar se basa en este paradigma, lo que implica que la evaluación de su mantenibilidad se debe centrar en la capacidad del código para ser modificado y mejorado a lo largo del tiempo, sin afectar su funcionamiento. Además, el enfoque OO permite una mayor modularidad y reutilización de código, lo que puede mejorar la mantenibilidad del producto a largo plazo [137], [138], [162], [163]. En el presente estudio, cuando se haga referencia a la mantenibilidad del software, se estará hablando específicamente sobre la mantenibilidad en sistemas OO.

La complejidad lógica del código fuente se correlaciona fuertemente con la mantenibilidad del software resultante [124], [128]. Diversos estudios en el ámbito de la IS han demostrado que la revisión de código es una de las técnicas más utilizadas para mejorar la mantenibilidad del software. En una revisión de literatura sistemática se encontró que esta práctica es muy extendida en la industria del software, con más del 70% de las organizaciones que la utilizan [164]. Asimismo, se ha demostrado que el análisis de métricas de código es ampliamente utilizada para medir y mejorar la mantenibilidad del software [165]. La mayoría de estás métricas están relacionadas directamente con sistemas OO [113].

En ese sentido, se había considerado realizar un estudio de mapeo sistemático sobre las métricas y modelos de calidad del producto de software, con el fin de evaluar su mantenibilidad y AF. Sin embargo, durante el proceso de investigación se encontró un estudio actual [113] que sirvió de referencia y base para la evaluación de estos atributos. Este estudio evaluó 70 artículos y congresos publicados entre 2009 y 2019, proporcionando información valiosa sobre las métricas y modelos utilizados para evaluar la calidad del producto del software.

Por consiguiente, una vez que los grupos han entregado los artefactos del producto de software, se procedió a realizar la evaluación de los atributos correspondientes, siguiendo el proceso indicado en la Fig. 9.

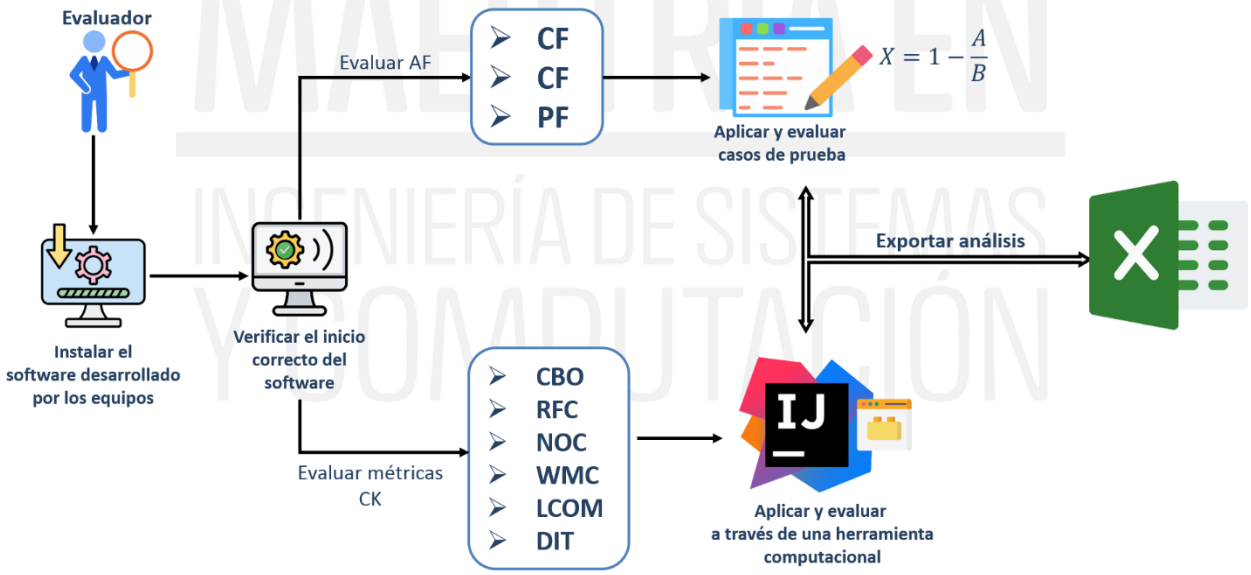


Fig. 9. Flujo para evaluar la mantenibilidad y adecuación funcional.

Fuente: esta investigación.

Es crucial preparar adecuadamente el entorno de prueba antes de proceder a la evaluación de la AF y la mantenibilidad del software. Esto implica instalar el software de forma local para asegurarse de que la prueba se realiza en un entorno controlado y reproducible. Si el entorno de prueba no está bien preparado, los resultados de las pruebas pueden ser incorrectos o inconsistentes. Por lo tanto, la instalación del software debe ser realizada siguiendo las especificaciones y requisitos necesarios, y se debe asegurar que el software funciona correctamente antes de comenzar las pruebas. Solo con un entorno de prueba bien preparado y una instalación de software adecuada se pueden obtener resultados precisos y fiables al evaluar dichos atributos.

A continuación, se explica los pasos para evaluar cada uno de los atributos mencionados.

a) Evaluación de la adecuación funcional

Para llevar a cabo esta evaluación de manera efectiva, es fundamental tener en cuenta los requisitos del usuario, que establecen las funcionalidades necesarias del sistema, y las historias de usuario que detallan cómo esas funcionalidades deben ser implementadas. A partir de estos elementos, se pudieron crear los casos de prueba funcional que permitieron verificar si el sistema cumple con los requerimientos establecidos por el usuario y si su funcionalidad es la esperada. Dichos casos de prueba se detallan en el Anexo 4.

Una vez gestionados los casos de prueba, se procedió a evaluar la completitud funcional, corrección funcional y pertinencia funcional, teniendo en cuenta la siguiente fórmula:

$$X = 1 - \frac{A}{B}$$

Completitud funcional (COMF):

La COMF se refiere a qué tan completo es el software en términos de las funciones que se esperan de él. Para evaluar esta métrica, se utilizó la fórmula mencionada anteriormente, en la que "A" representa el número total de funciones faltantes y "B" representa el número total de funciones especificadas.

Corrección funcional (CORF):

La CORF se refiere a qué tan bien el software realiza las funciones que se esperan de él sin errores o fallas. Para evaluar esta métrica, se utilizó la fórmula mencionada anteriormente, en la que "A" representa el número total de funciones incorrectas, mientras que "B" representa el número total de funciones implementadas.

Pertinencia funcional (PERF):

La PERF se refiere a qué tan bien el software satisface las necesidades y expectativas del usuario. Para evaluar esta métrica, se utilizó la fórmula mencionada anteriormente, en la que "A" representa el número total de funciones adecuadas para lograr los objetivos del usuario, mientras que "B" representa el número total de funciones implementadas.

Es importante tener en cuenta que, entre más cercano a 1 sea el puntaje es más favorable. Es decir, mejor será la evaluación del software en cuanto a la implementación de las funciones requeridas [150],[54],[149],[151].

Los resultados de la evaluación de la AF pueden presentarse de manera clara y ordenada en una hoja de cálculo de Excel, en la cual se registraron los resultados de la evaluación para cada métrica. En la TABLA VI **¡Error! No se encuentra el origen de la referencia.**se presenta la tabulación sugerida de resultados para la AF.

TABLA VII
TABULACIÓN SUGERIDA DE RESULTADOS PARA LA ADECUACIÓN FUNCIONAL

Grupo	COMF	CORF	PER
G1	0.69	0.90	0.80
.	.	.	.
.	.	.	.
.	.	.	.
Promedio	0.70	0.80	0.60

Fuente: esta investigación.

Combinando la evaluación cualitativa con la evaluación cuantitativa, se puede obtener una visión completa del rendimiento del software en términos de AF. La evaluación cualitativa puede identificar problemas específicos del software y proporcionar información detallada sobre la calidad del software, mientras que la evaluación cuantitativa puede proporcionar una visión general del rendimiento del software en términos de cumplimiento de requisitos del usuario. En conjunto, la evaluación cualitativa y cuantitativa pueden ayudar a mejorar el software y garantizar que cumpla con las necesidades y expectativas del usuario.

Al evaluar la AF de un software, es importante tener en cuenta que pueden existir sesgos, ya que estas evaluaciones a menudo se basan en opiniones subjetivas. Una forma de disminuir este sesgo es realizar la evaluación entre dos personas o más personas que tengan conocimientos relevantes en el contexto del desarrollo de software. De esta forma, se pueden obtener diferentes perspectivas y puntos de vista, lo que puede ayudar a reducir la influencia de los sesgos individuales. Además, es importante utilizar un criterio claro y bien definido para evaluar la AF del software y registrar los resultados de manera detallada para poder justificar y explicar cualquier puntuación o calificación asignada. Esto ayudó a garantizar que la evaluación sea lo más objetiva y justa posible.

b) Evaluar la mantenibilidad

La evaluación de la mantenibilidad en sistemas OO es fundamental para garantizar la calidad y la facilidad de mantenimiento del software [49], [144]. Para facilitar el análisis y la evaluación de la mantenibilidad, se han utilizado en gran medida las métricas de Chidamber y Kemerer (CK) [23], [49], [113], [142]–[144], [166], [167]. Las cuales se basan en seis características de los sistemas OO, que incluyen la complejidad, cohesión, acoplamiento, herencia, polimorfismo y encapsulamiento [49], [129].

Para obtener los resultados de las métricas CK, se pueden utilizar diferentes herramientas de análisis estático de código. Una herramienta popular para este propósito es el plugin MetricsReloaded, que se puede integrar con la herramienta IntelliJ IDEA Community Edition [168]. Este plugin permitió obtener los valores de las métricas CK para cada clase del sistema y generar informes que indican el nivel de cada clase.

Para evaluar las métricas CK con la herramienta IntelliJ IDEA Community Edition, es necesario importar de forma individual el proyecto desarrollado por los grupos. Una vez importado, se debe

ejecutar el plugin correspondiente, el cual inicia el análisis y mostrará los resultados en una ventana, tal como se indica en la Fig. 10**¡Error! No se encuentra el origen de la referencia.**.

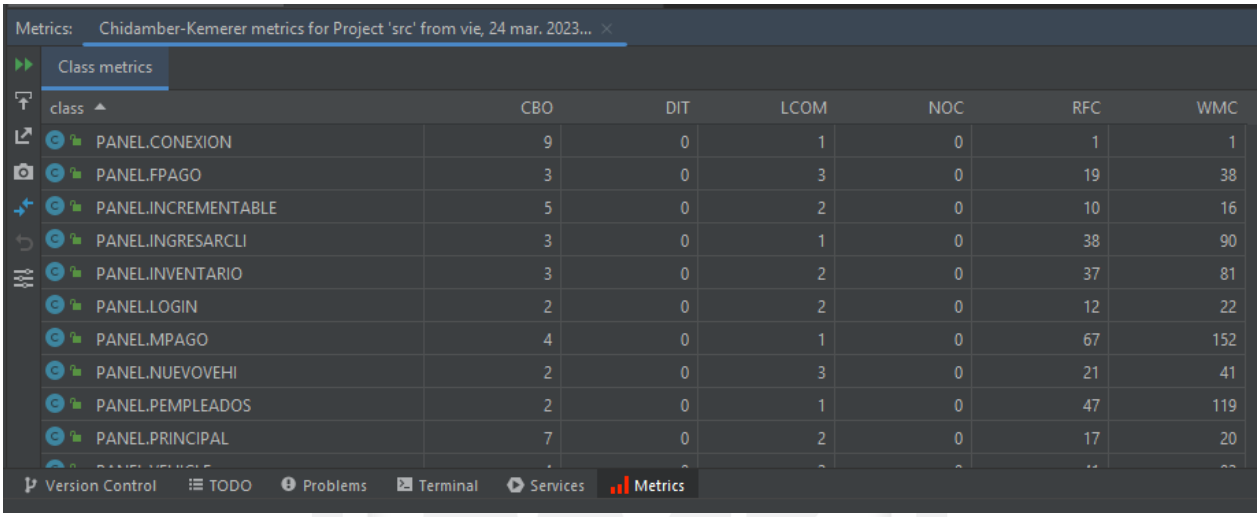


Fig. 10. Evaluación de prueba en IntelliJ IDEA para las métricas CK.

Fuente: esta investigación.

En la figura anterior se muestra los valores de las seis métricas CK (CBO, DIT, LCOM, NOC, RFC y WMC) para cada una de las clases del proyecto analizado. Cada fila representa una clase diferente y las columnas muestran los valores de las diferentes métricas CK para cada clase. Además, en la parte final de la tabla se incluye un promedio de los valores de las métricas para todas las clases. Es importante destacar que este proceso se debe realizar con todos los proyectos desarrollados por los grupos. Además, se debe contar con una estructura clara y estandarizada para la presentación de los resultados de la evaluación de las métricas CK de los diferentes grupos. La TABLA VIII proporciona una tabulación sugerida para este propósito, permitiendo una comparación efectiva y un análisis detallado de los resultados obtenidos por cada grupo.

TABLA VIII
TABULACIÓN SUGERIDA PARA LOS RESULTADOS DE LA EVALUACIÓN DE LAS MÉTRICAS CK

Grupo	CBO	DIT	LCOM	NOC	RFC	WMC
GI1	4,00	1,00	1,82	3,00	28,18	61,09
GI2	2,62	1,00	2,92	2,00	17,31	27,19
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
Promedio	3,71	1,02	2,10	0,92	12,19	20,49

Fuente: esta investigación.

5) Evaluar y contrastar los resultados obtenidos por los grupos

Una vez que se ha llevado a cabo la recopilación y organización de la información en las tablas correspondientes, es necesario proceder a evaluar y contrastar los resultados obtenidos por el grupo experimental y de control utilizando técnicas estadísticas. Estas técnicas estadísticas permitieron realizar un análisis correlacional para determinar si existe una relación entre las variables medidas.

Además, se utilizan para determinar si las hipótesis planteadas se cumplen, lo cual es esencial para validar los resultados de la investigación. Entre las técnicas estadísticas utilizadas en el presente estudio se encuentran el test de normalidad de Shapiro-Wilk y las pruebas U de Mann-Whitney y t de Student. El test de normalidad de Shapiro-Wilk permite verificar si los datos siguen una distribución normal, mientras que las pruebas U de Mann-Whitney y t de Student se utilizaron para comparar las diferencias entre los grupos y determinar si estas diferencias son estadísticamente significativas. El uso de estas técnicas estadísticas aseguró una evaluación rigurosa de los datos y permite obtener conclusiones significativas y relevantes en la investigación científica.

C. VARIABLES E INDICADORES

La variable independiente son los tipos de formación de los grupos, una variable nominal con dos niveles, grupos homogéneos bajo la dimensión de la meticulosidad (GHBDM) y grupos formados por preferencia de los estudiantes (GFPE). El grupo de control corresponde a GFPE y el grupo experimental a GHBDM.

Por otro lado, se considera un total de nueve variables dependientes, seis de ellas son: WMC (Weighted Methods per Class), DIT (Depth of Inheritance Tree), NOC (Number of Children), CBO (Coupling Between Objects), RFC (Response for a Class) y LCOM (Lack of Cohesion in Methods). La cuales corresponden a las métricas CK, estas variables son útiles para evaluar diferentes aspectos del diseño de un sistema OO que pueden influir en la mantenibilidad del mismo. A demás, la TABLA IX**¡Error! No se encuentra el origen de la referencia.** indica los umbrales recomendados y deseados para cada variable en términos de mantenibilidad del software, los cuales fueron propuestos por M. Manso et al. [49], guiados por recomendaciones de la NASA [144].

TABLA IX
UMBRALES RECOMENDADOS Y DESEADOS PARA LAS MÉTRICAS CK

Métrica	Umbral
WMC	5 a 10 (influencia en la mantenibilidad es del 85% a 67%)
DIT	1 a 3 (99% a 74%)
NOC	2 a 4 (70% a 10%)
CBO	1 a 3 (92% a 86%)
RFC	5 a 10 (96% a 84%)
LCOM	0.05 a 0.18 (94% a 75%)

Fuente: esta investigación.

Las tres variables restantes son: Completitud Funcional (COMF), Corrección Funcional (CORF) y Pertinencia Funcional (PERF). Las cuales corresponden al constructo AF, tal como se describe en la sección de supuestos teóricos.

Con el fin de asegurar la validez del experimento, se controlaron o bloquearon las variables independientes, manteniendo constantes las condiciones en las que se desarrolló el estudio. De esta manera, se buscó minimizar el impacto de factores externos que pudieran interferir en la relación entre la variable independiente y las variables dependientes. En este sentido, se aseguró que todas las variables que no fueron manipuladas tuvieran el mismo nivel. Las variables que se mantuvieron constantes durante el experimento fueron las siguientes:

- Experiencia del estudiante en el test psicológico aplicado: aunque todos confirmaron que nunca habían realizado un test psicológico; se les recordó que no hay respuestas correctas o incorrectas y que deben ser precisas y veraces.

- Experiencia de los participantes como programadores: factor indeseable que pueden influir en la calidad del software desarrollado, a través de una sesión con los estudiantes, se interrogó si alguno anteriormente había trabajo en la industria y/o tenía algún tipo de experiencia demostrable en el desarrollo de software. Todos los alumnos respondieron que no tenían ningún tipo experiencia, es decir todos eran programadores novatos.
- Producto software a desarrollar: funcionalidades que los participantes deberán implementar durante el experimento. Si los grupos trabajan en diferentes problemas y lenguajes de programación, es posible que los resultados de los grupos no sean comparables. Esta variable se bloquea al hacer que todos los grupos trabajen en el mismo proyecto y en el lenguaje de programación JAVA.

En la TABLA X se presenta la Sistematización de variables de la investigación.

TABLA X
SISTEMATIZACIÓN DE VARIABLES DE LA INVESTIGACIÓN

Variable	Tipo de formación de los grupos	
Descripción	Una variable nominal con dos niveles: grupos homogéneos bajo la dimensión de la meticulosidad y grupos formados por preferencia de los estudiantes. Evaluar la personalidad es un insumo relevante para formas los grupos homogéneos.	
Tipo de Variable	Independiente	
Objetivo específico	Caracterizar los modelos psicológicos empleados para identificar los rasgos de personalidad de los desarrolladores de software.	
	Indicador	Naturaleza
	Actividad	Cualitativa
	Flujo de control	Cualitativa
	Actor	Cualitativa
Fuente	Base de datos bibliográfica	
Técnica de recolección	Revisión documental	
Técnica de análisis	Análisis documental	
Variable	Mantenibilidad y adecuación funcional	
Descripción	Métricas para evaluar la mantenibilidad y la adecuación funcional del producto software	
Tipo de Variable	Dependiente	
Objetivo específico	Caracterizar los modelos y las métricas de calidad del producto software para evaluar la adecuación funcional y la mantenibilidad del producto software.	
	Indicador	Naturaleza
	Actividad	Cualitativa
	Flujo de control	Cualitativa

	Actor	Cualitativa
Fuente	Base de datos bibliográfica	
Técnica de recolección	Revisión documental	
Técnica de análisis	Análisis documental	

Fuente: esta investigación.

D. POBLACIÓN Y MUESTRA

Los participantes fueron estudiantes de quinto semestre del Programa de Ingeniería de Sistemas de la Universidad de Nariño (sede Pasto e Ipiales), Colombia. Estos alumnos estaban cursando la asignatura de Ingeniería de Software III.

El experimento involucró un total de 76 participantes, de los cuales 30 correspondieron a estudiantes de la sede Ipiales (grupo experimental). Dentro de este grupo, 6 fueron mujeres (20%) y 24 fueron hombres (80%). Se formaron 10 grupos en total, cada uno compuesto por tres miembros.

Por otra parte, 46 estudiantes correspondieron a la sede Pasto (grupo de control), de los cuales, 42 fueron hombres (91,30%) y 4 fueron mujeres (8,70%). En total se formaron 9 grupos, 8 de ellos estaban compuestos por 5 miembros y 1 por 6 miembros. La elección de formar grupos de 5 personas en el grupo de control fue motivada por razones prácticas. Debido a la gran cantidad de estudiantes en la sede Pasto, el tiempo disponible para la evaluación era limitado y la evaluación individual de cada uno de ellos por parte del docente podía resultar en un proceso largo y tedioso. Al formar grupos de 5 personas, se pudo reducir el número de evaluaciones necesarias, lo que permitió al docente dedicar más tiempo a cada evaluación individual y obtener resultados más precisos en menos tiempo.

E. INSTRUMENTOS DE RECOLECCIÓN DE INFORMACIÓN

En el presente estudio, se emplearon los siguientes instrumentos para recopilar información:

Cuestionario BFI

Aunque la recopilación de información sobre los rasgos de personalidad de los estudiantes no forma parte del análisis estadístico en el presente estudio, este instrumento se incluye a modo de referencia para demostrar cómo se obtuvieron estos rasgos de personalidad.

Dicho lo anterior, se empleó el cuestionario BFI (Inventario de los Cinco Grandes) para evaluar los rasgos de personalidad de los estudiantes. Este instrumento consta de 44 preguntas de respuesta múltiple tipo Likert que miden las cinco dimensiones fundamentales del modelo Big Five: Extroversión, Amabilidad, Meticulosidad, Neuroticismo y Apertura a Nuevas Experiencias. En la**¡Error! No se encuentra el origen de la referencia.** TABLA IV se muestran estos ítems organizados por cada una de las dimensiones. Para este estudio, se utilizó la versión en español del BFI desarrollada por John et al. [83], conocida como “BFI español” (ver Fig. 11).

Casos de prueba para evaluar la AF

Los casos de prueba funcional son una herramienta clave en la recolección de información en el ámbito del desarrollo de software. Estos casos permiten evaluar el comportamiento del sistema en diferentes situaciones y escenarios, ayudando a identificar posibles errores o fallos en su

funcionamiento. Además, los casos de prueba funcional también permiten evaluar la calidad y eficacia del software, asegurando que cumpla con los requerimientos y expectativas de los usuarios. Por lo tanto, los casos de prueba funcional son un instrumento esencial en la recolección de información para el desarrollo de software y en la garantía de su calidad y eficacia [169].

MetricsReload para la recopilación de las métricas CK

Para recolectar las métricas, se realizó una búsqueda para encontrar una herramienta que realizara esta tarea automáticamente y obedeciera a algunos criterios, tales como: ser un software simple de usar, no comercial y que pudiera recolectar métricas de forma confiable.

MetricsReload es una herramienta complementaria que se integra con IntelliJ IDEA Community Edition para facilitar la recopilación de métricas CK. Esta herramienta se puede descargar e instalar como un plugin a través del sitio web oficial de JetBrains (https://plugins.jetbrains.com/plugin/93-metricsreloaded). Al utilizar MetricsReload, se pueden obtener métricas detalladas sobre la complejidad, cohesión, acoplamiento y otros aspectos importantes del código de manera fácil y rápida, lo que reduce la necesidad de recopilar los datos manualmente. Además, MetricsReload presenta la información de manera clara y sencilla a través de su interfaz gráfica de usuario intuitiva, lo que facilita la comprensión de los datos recopilados y su posterior análisis.

F. VALIDEZ Y CONFIABILIDAD DE LOS INSTRUMENTOS

Cuestionario BFI (Inventario de los Cinco Grandes)

BFI es ampliamente utilizado para evaluar la personalidad. Este instrumento se basa en la teoría de los cinco grandes rasgos de la personalidad (Big Five): Extraversión, Amabilidad, Meticulosidad Neuroticismo y Apertura a Nuevas Experiencias. Hay varios argumentos que respaldan la validez de la aplicabilidad de BFI para evaluar la personalidad y su eficacia para predecir el comportamiento de los fenómenos que se estudian [83].

Primero, el instrumento BFI tienen una amplia aceptación en la comunidad científica y han sido validados en numerosos estudios empíricos. Estos estudios han demostrado que los resultados obtenidos a través de los instrumentos de cuestionario BFI son consistentes y precisos. Además, los resultados obtenidos a través de estos instrumentos han demostrado ser predictivos de una variedad de resultados, incluyendo comportamientos, emociones y cogniciones [170].

En segundo lugar, el instrumento tiene una estructura clara y bien definida. Los cinco grandes rasgos de la personalidad son dimensiones estables y amplias que cubren una amplia gama de comportamientos y características. Estos rasgos son independientes entre sí, lo que significa que la medición de un rasgo no afecta la medición de los demás. La estructura clara y bien definida de los instrumentos de cuestionario BFI ayuda a garantizar que se mida lo que realmente se desea medir y que los resultados sean precisos y confiables [171].

En tercer lugar, el cuestionario tiene una alta consistencia interna. Esto significa que las preguntas en el cuestionario están altamente correlacionadas con el rasgo que se está midiendo. Por lo tanto, si se vuelve a aplicar el cuestionario, los resultados deberían ser muy parecidos o similares. La alta consistencia interna de los instrumentos de cuestionario BFI ayuda a garantizar que los resultados sean confiables y precisos [83].

El cuestionario puede ser utilizado en una variedad de contextos y para una variedad de propósitos. Además de evaluar la personalidad, los instrumentos de cuestionario BFI también pueden ser

utilizados para evaluar la AF en entornos laborales y educativos. Estos instrumentos pueden ayudar a identificar las fortalezas y debilidades de un individuo y proporcionar información valiosa para la toma de decisiones [170].

Por último, en el ámbito de la IS el modelo Big Five, acompañado de sus correspondientes instrumentos de medición como el BFI, ha ganado gran importancia y aceptación entre los investigadores para la evaluación de la personalidad de los desarrolladores de software [48].

Casos de prueba para evaluar la AF

Los casos de prueba son una herramienta importante para evaluar la AF del software. A continuación, se presentan algunos argumentos que respaldan su validez y aplicabilidad [2], [122], [172]:

- Los casos de prueba son una forma sistemática y estructurada de probar el software. Al seguir un conjunto de casos de prueba, es posible cubrir todas las funcionalidades del software y garantizar que se han probado todos los escenarios posibles.
- Los casos de prueba permiten evaluar la calidad del software en términos de su capacidad para satisfacer los requisitos funcionales establecidos. Al probar el software con casos de prueba, se pueden identificar las áreas donde el software no cumple con los requisitos y se pueden tomar medidas para corregir los problemas.
- Los casos de prueba permiten evaluar la capacidad del software para manejar situaciones excepcionales. Los casos de prueba pueden incluir escenarios que simulen situaciones inesperadas o extremas, como errores de entrada de datos o problemas de conexión a la red. Al probar el software con estos casos de prueba, se puede evaluar su capacidad para manejar estas situaciones.
- Los casos de prueba son repetibles y consistentes. Si se ejecutan los mismos casos de prueba en varias ocasiones, se deben obtener resultados similares. Esto garantiza que los resultados sean confiables y que se puedan detectar cambios en el comportamiento del software con el tiempo.

MetricsReload para la recopilación de las métricas CK

IntelliJ IDEA Community Edition como MetricsReload son herramientas ampliamente utilizadas tanto en la industria como en investigaciones relacionadas con el desarrollo de software, lo que demuestra su eficacia y fiabilidad en la medición de métricas CK y su relevancia en el campo de la IS [23], [163]. Es importante destacar que tanto IntelliJ IDEA Community Edition como su complemento MetricsReload son herramientas gratuitas y de fácil acceso, lo que ha contribuido a su amplia utilización por parte de la comunidad de desarrolladores. Además, MetricsReload cuenta con más de 195,000 descargas [173], lo que la convierte en una de las herramientas más utilizadas en comparación con otras. La comunidad de usuarios también ha desarrollado una gran cantidad de documentación y soporte para estas herramientas, lo que indica que han sido probadas y refinadas con el tiempo [168].

Por lo tanto, IntelliJ IDEA como MetricsReload son herramientas precisas y confiables en la recopilación de métricas CK. Esto se debe a que estas métricas se basan en la estructura del código fuente y la complejidad, lo que hace poco probable que los resultados varíen significativamente entre diferentes ejecuciones de las herramientas en el mismo código fuente. Por lo tanto, es posible garantizar la consistencia y exactitud de los resultados obtenidos a través de estas herramientas.

Además, el uso de estas herramientas ayuda a reducir el error humano al recopilar datos manualmente, lo que aumenta aún más la fiabilidad de los resultados obtenidos.

G. PROCESAMIENTO DE LA INFORMACIÓN

El proceso de recopilación y análisis de datos se llevó a cabo mediante el uso de diferentes herramientas que permitieron la eficiente tabulación y procesamiento de la información. Entre estas herramientas destacan Excel y SPSS™ [174], que fueron utilizados para la consolidación y análisis de los resultados de las variables dependientes tanto de la mantenibilidad como la AF.

Estas herramientas permitieron la creación de tablas y gráficos que facilitaron la comprensión de los datos recopilados y permitieron la identificación de patrones y tendencias en los mismos. Además, el uso de estas herramientas contribuyó a garantizar la precisión y la confiabilidad de los resultados obtenidos, al proporcionar herramientas de análisis estadístico robustas y bien establecidas en el campo de la investigación científica.

Para procesar la información de las variables dependientes en relación a la mantenibilidad, es necesario en primer lugar evaluar el código del producto desarrollado por los grupos utilizando la herramienta IntelliJ IDEA, la cual genera los resultados promedio por cada métrica, como se puede observar en la Fig. 10. Estos resultados promedios deben ser almacenados en una hoja de cálculo, manteniendo la tabulación sugerida en la TABLA VIII para facilitar la organización y comprensión de los resultados por cada grupo. A continuación, se deben agrupar los resultados promedios tanto del grupo de control como experimental. Para analizar los promedios de cada variable de la mantenibilidad generados por los grupos, se pueden utilizar visualizaciones gráficas en Excel, como gráficos de barras o gráficos circulares para presentar los resultados de manera clara y concisa.

Para procesar la información de las variables dependientes en relación a la AF, es esencial tener definidos los casos de prueba y ejecutarlos de manera adecuada. A través de las fórmulas presentadas en el punto 4 de la sección B, se pueden obtener los puntajes de las métricas de COMF, CORF y PERF por cada grupo. Estos resultados son almacenados en una hoja de cálculo, manteniendo la tabulación sugerida en la TABLA VII para facilitar la organización y comprensión de los resultados por cada grupo. A continuación, se deben agrupar los resultados promedios tanto del grupo de control como experimental. Para analizar los promedios de cada métrica generados por los grupos, es recomendable utilizar visualizaciones gráficas en Excel, como gráficos de barras o gráficos circulares, para presentar los resultados de manera clara y concisa. De esta manera, se logra procesar eficientemente la información de dichas variables y obtener una comprensión detallada de su relación con el desarrollo de software por cada grupo.

Para obtener información detallada sobre el procesamiento de datos y el análisis de resultados para las anteriores variables dependientes, se puede acceder al siguiente enlace https://onx.la/742cd.

Por otro lado, se utilizó la herramienta IBM SPSS para llevar a cabo el contraste de hipótesis y analizar la relación entre las diferentes variables en el conjunto de datos. Gracias a esta herramienta, se obtuvo una comprensión más detallada de las relaciones estadísticas entre el grupo experimental y de control con respecto a las variables dependientes, lo que resultó en resultados significativos y relevantes para el proyecto.

Se utilizó la prueba de normalidad de Shapiro-Wilk para determinar si los datos seguían una distribución normal. Esta evaluación es importante para determinar qué técnica de contraste de

hipótesis se debe utilizar. Si los datos no siguen una distribución normal, se debe utilizar la prueba U de Mann-Whitney, mientras que si los datos son normales, se debe utilizar la prueba t de Student.

Para obtener información detallada sobre el procesamiento de datos y el análisis de resultados para las variables utilizando SPSS, se puede acceder al siguiente enlace https://onx.la/fd796.

IV. RESULTADOS DE LA INVESTIGACIÓN

En esta sección, se presenta los resultados del estudio realizado. En una primera instancia, se expondrán los resultados relacionados con los objetivos específicos que se plantearon en este trabajo de investigación. Estos resultados estarán enfocados en las respuestas que se encontraron para cada uno de los objetivos específicos y en cómo estos contribuyen al logro del objetivo general del estudio. Posteriormente, se presentarán los resultados estadísticos obtenidos en el análisis de los datos, los cuales permiten establecer el grado de relación entre las variables estudiadas. Todos los resultados serán analizados y discutidos con detalle con el objetivo de brindar una interpretación rigurosa y precisa de los hallazgos obtenidos.

Resultados del estudio en relación a los objetivos específicos.

El primer objetivo planteado en este estudio tuvo como resultado la presentación de una ponencia en el VIII Congreso Iberoamericano de Interacción Humano Computador realizado en Cuba. Esta ponencia se enfocó en los resultados obtenidos a través del análisis de los datos recolectados y su relación con el objetivo general del estudio. Además, el artículo resultó seleccionado para ser publicado en el libro HCI-COLLAB 2022 de la editorial Springer dentro de la serie CCIS. Este logro es una muestra del valor y la relevancia de los hallazgos obtenidos en este estudio y de su aporte al avance del conocimiento en el campo de la IS. Los detalles sobre la ponencia y la publicación del artículo se presentan en el Anexo 5 y Anexo 6 respetivamente.

El segundo objetivo específico de este estudio tenía como finalidad realizar un mapeo sistemático para caracterizar los modelos y métricas que permiten evaluar el producto de software. Durante el proceso de investigación se encontró un estudio actual [113] que sirvió como referencia y base importante para evaluar los atributos de mantenibilidad en sistemas orientados a objetos y la AF.

Este estudio evaluó 70 artículos y congresos publicados entre 2009 y 2019, proporcionando información valiosa sobre las métricas y modelos utilizados para evaluar la calidad del producto de software. Como resultado, se identificaron diversas métricas para evaluar la calidad del producto de software. Entre ellas se encuentran las métricas de Chidamber y Kemerer (CK), las cuales han sido evaluadas y aplicadas en múltiples estudios empíricos. Estas métricas son consideradas esenciales para evaluar la calidad y mantenibilidad del software OO.

Con respecto a la AF, es un aspecto crítico de la calidad del software, su evaluación es compleja debido a la variabilidad inherente en los requisitos funcionales de cada sistema, lo que hace que no existan métricas específicas para su evaluación. Sin embargo, el estudio de referencia permitió destacar que el modelo de calidad ISO 25010 es un marco de referencia útil para evaluar la AF del software. El modelo de calidad incluye tres subcaracterísticas específicas para evaluar la AF: Completitud, Corrección y Pertinencia funcional. En general, la evaluación de la AF se basa en pruebas exhaustivas para verificar que el software cumpla con todos los requisitos funcionales especificados.

Finalmente, como resultado del tercer objetivo específico, se logró consolidar un artículo científico. Actualmente, el artículo se encuentra publicado en la Revista Ciencia e Ingeniería Neogranadina. La publicación de los resultados de la investigación, permitirá compartir los hallazgos con la comunidad académica y científica, fomentando la ampliación y mejora de los mismos por parte de otros investigadores. Los detalles sobre la publicación del artículo se presentan en el Anexo 7.

Resultados estadísticos.

Al finalizar el experimento se aplicó a los grupos de estudio una post-prueba (O_1 y O_2), cuyo objetivo fue determinar la implicación del tratamiento experimental. Esta post-prueba consistió en la evaluación de las variables dependientes relacionadas con la mantenibilidad, como WMC (Weighted Methods per Class), DIT (Depth of Inheritance Tree), NOC (Number of Children), CBO (Coupling Between Objects), RFC (Response for a Class) y LCOM (Lack of Cohesion in Methods). Además, se evaluaron la Completitud Funcional (COMF), Corrección Funcional (CORF) y Pertinencia Funcional (PERF), que miden la adecuación funcional del sistema. A través de la post-prueba, permitió contrastar el grupo experimental versus el grupo de control, buscando determinar si existe una mejora con respecto a las variables dependientes por parte de grupos homogéneos bajo la dimensión de meticulosidad frente a grupos formados por preferencia de los estudiantes.

Estadística descriptiva para las variables dependientes

A continuación, se presentan los resultados mediante la estadística descriptiva en forma de tablas y gráficos. Los resultados obtenidos facilitan la comparación entre los distintos grupos y proporcionan una comprensión visual rápida y clara de los datos.

En la TABLA XI**¡Error! No se encuentra el origen de la referencia.** se muestra la puntuación de los grupos pertenecientes al grupo experimental en relación a las variables dependientes que corresponden a la mantenibilidad, mientras que la Fig. 11 presenta el contraste de las puntuaciones obtenidas por los grupos mencionados anteriormente.

TABLA XI
PUNTUACIONES DE LAS VARIABLES DE MANTENIBILIDAD DEL GE

Grupos	CBO	DIT	LCOM	NOC	RFC	WMC
GI1	4,00	0,00	1,82	0,00	28,18	61,09
GI2	2,62	0,00	2,92	0,00	17,31	27,19
GI3	7,87	0,11	1,97	0,11	6,97	11,62
GI4	2,46	0,00	1,46	0,00	5,15	14,00
GI5	2,31	0,00	2,00	0,00	13,31	24,77
GI6	4,00	0,05	2,97	0,05	13,44	15,51
GI7	2,89	0,00	2,39	0,00	11,94	16,61
GI8	2,69	0,00	2,34	0,00	7,83	10,26
GI9	3,80	0,00	1,05	0,00	9,35	12,75
GI10	4,43	0,00	2,09	0,00	8,43	11,11

Fuente: esta investigación.

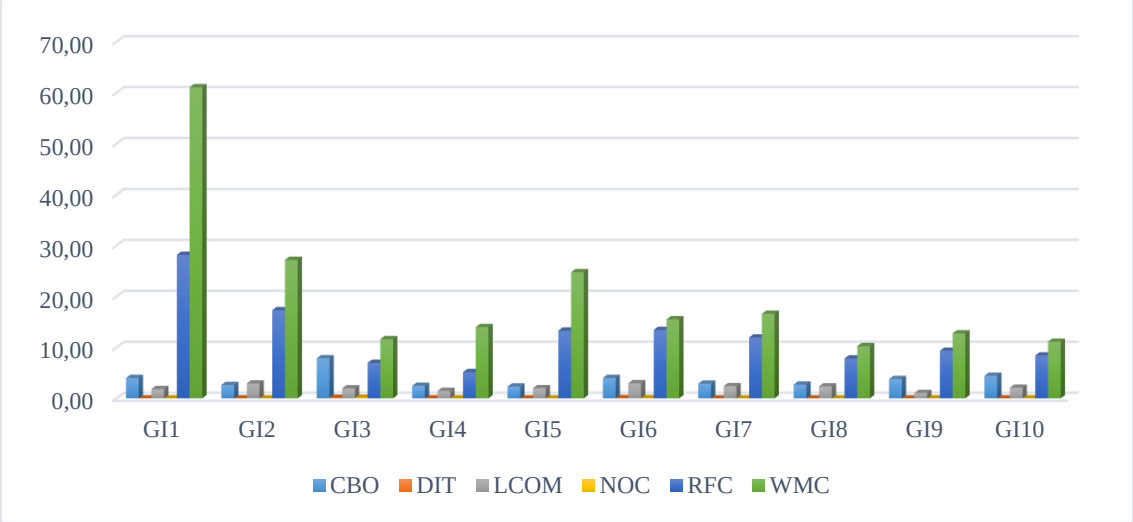


Fig. 11. Contraste de puntuaciones de las variables de mantenibilidad del grupo experimental.

Fuente: esta investigación.

En la TABLA XII**¡Error! No se encuentra el origen de la referencia.** se muestra la puntuación de los grupos pertenecientes al grupo de control en relación a las variables dependientes que corresponden a la mantenibilidad, mientras que la Fig. 12 presenta el contraste entre las puntuaciones obtenidas por los grupos mencionados anteriormente.

TABLA XII
PUNTUACIONES DE LAS VARIABLES DE MANTENIBILIDAD DEL GC

Grupos	CBO	DIT	LCOM	NOC	RFC	WMC
GP1	3,39	0,00	1,61	0,00	9,83	10,17
GP2	7,88	0,00	2,88	0,00	28,47	34,24
GP3	3,29	0,00	3,03	0,00	11,38	9,82
GP4	3,33	0,00	1,56	0,00	17,33	34,11
GP5	3,43	0,00	1,07	0,00	8,36	12,86

Fuente: esta investigación.

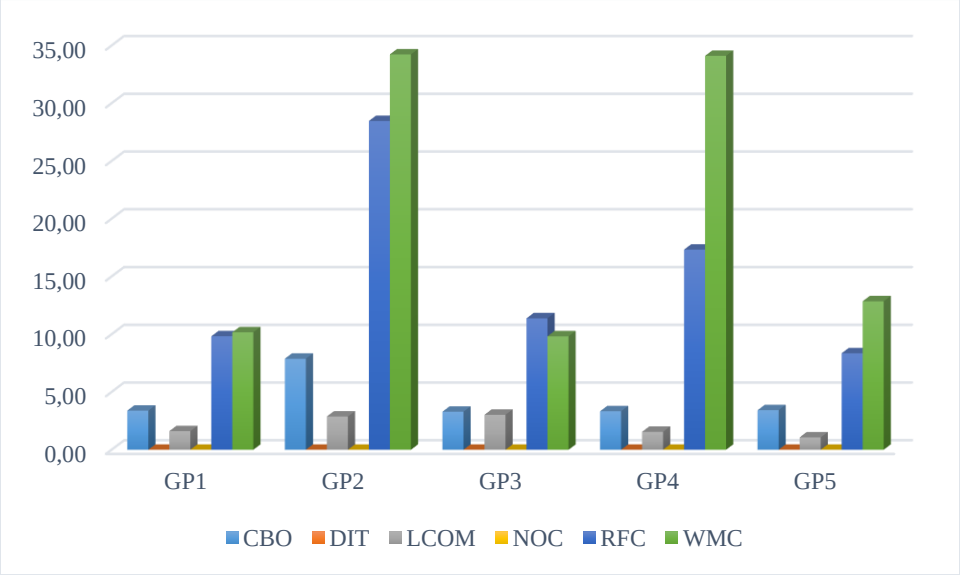


Fig. 12. Contraste de puntuaciones de las variables de mantenibilidad del grupo de control.

Fuente: esta investigación.

La TABLA XIII**¡Error! No se encuentra el origen de la referencia.** muestra la puntuaciones medias de las variables dependientes correspondientes a la mantenibilidad tanto para el grupo experimental como para el grupo de control, mientras que la Fig. 13 muestra el contraste de estas puntuaciones obtenidas en cada uno de los agrupamientos considerados. Los resultados indican que, en promedio, las puntuaciones obtenidas en la post-prueba por el grupo experimental son aparentemente similares a las obtenidas por el grupo de control en relación a las variables dependientes que corresponden a la mantenibilidad.

TABLA XIII
PUNTUACIONES MEDIAS DE LAS VARIABLES DE MANTENIBILIDAD (GE VS GC)

Grupos	CBO	DIT	LCOM	NOC	RFC	WMC
Experimental	3,71	0,02	2,10	0,02	12,19	20,49
Control	4,27	0,00	2,03	0,00	15,07	20,24

Fuente: esta investigación.

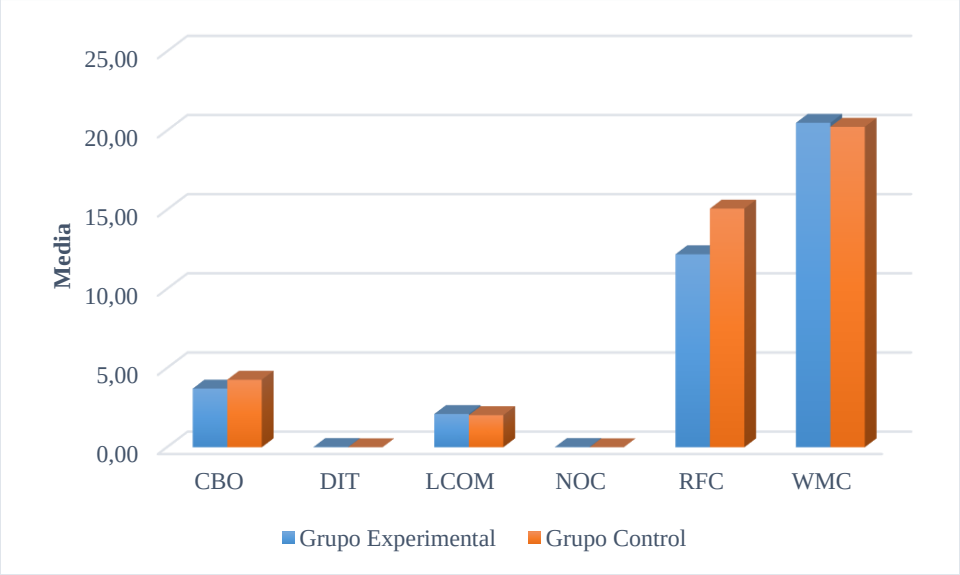


Fig. 13. Contraste de puntuaciones de las variables de mantenibilidad, grupo experimental vs grupo de control.

Fuente: esta investigación.

Po otro lado, en la TABLA XIV**¡Error! No se encuentra el origen de la referencia.** se presenta la puntuación de los grupos pertenecientes al grupo experimental en relación a las variables dependientes que corresponden a la AF, mientras que la Fig. 14 se presenta el contraste de las puntuaciones obtenidas por los grupos mencionados anteriormente.

TABLA XIV
PUNTUACIONES DE LAS VARIABLES DE LA AF DEL GE

Grupos	COMF	CORF	PERF
GI1	0,43	0,77	0,70
GI2	0,45	0,96	0,52
GI3	0,76	0,87	0,85
GI4	0,37	1,00	0,63
GI5	0,43	0,77	0,55
GI6	0,65	0,82	0,70
GI7	0,43	0,55	0,45
GI8	0,45	0,48	0,43
GI9	0,39	0,85	0,85
GI10	0,45	0,91	0,61

Fuente: esta investigación.

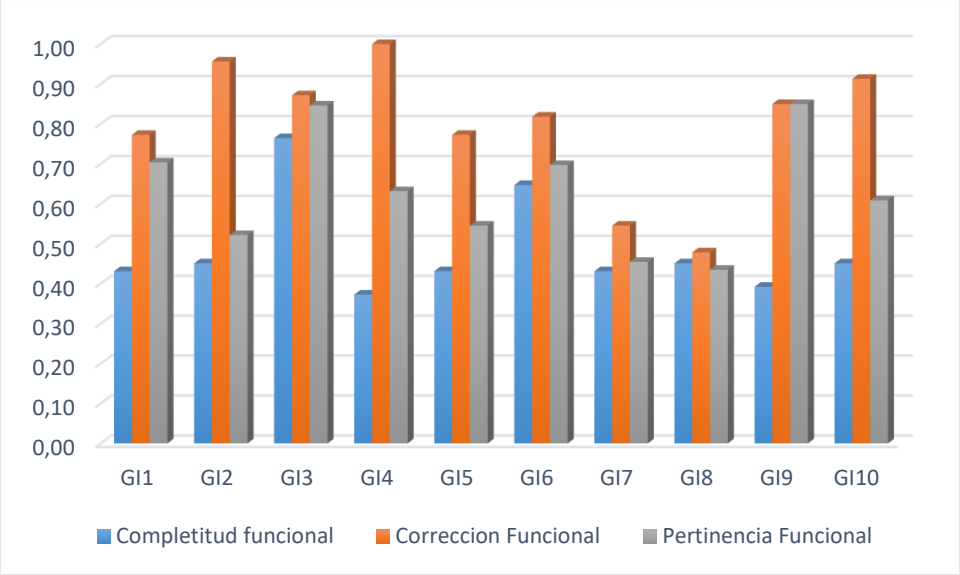


Fig. 14. Contraste de puntuaciones de las variables de la AF del grupo experimental.

Fuente: esta investigación.

En la TABLA XV**¡Error! No se encuentra el origen de la referencia.** se muestra la puntuación de los grupos pertenecientes al grupo de control en relación a las variables dependientes que corresponden a la AF, mientras que la Fig. 15 se presenta el contraste de las puntuaciones obtenidas por los grupos mencionados anteriormente.

TABLA XV
PUNTUACIONES DE LAS VARIABLES DE LA AF DEL GC

Grupos	COMF	CORF	PERF
GP1	0,39	0,70	0,40
GP2	0,76	0,77	0,62
GP3	0,59	0,73	0,74
GP4	0,29	0,87	0,53
GP5	0,57	0,83	0,59
GP6	0,25	0,85	0,77
GP7	0,75	0,90	0,59
GP8	0,37	0,84	0,82

Fuente: esta investigación.

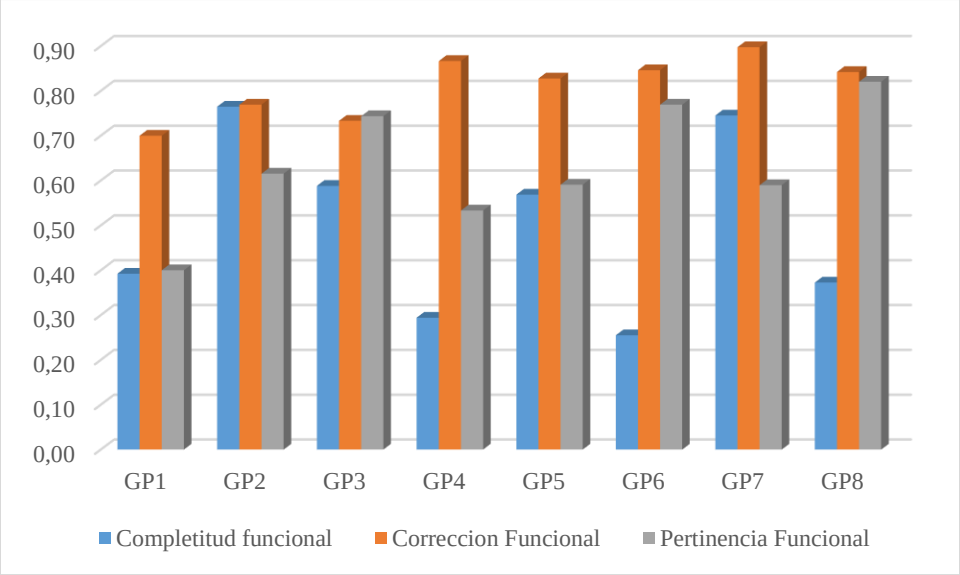


Fig. 15. Contraste de puntuaciones de las variables de la AF del grupo de control.

Fuente: esta investigación.

Finalmente, la TABLA XVI **¡Error! No se encuentra el origen de la referencia.**muestra las puntuaciones medias de las variables dependientes correspondientes a la AF tanto del grupo experimental como del grupo de control, mientras que la Fig. 16 muestra el contraste de estas puntuaciones obtenidas en cada uno de los agrupamientos considerados. Los resultados también indican que, en promedio, las puntuaciones obtenidas en la post-prueba por el grupo experimental son aparentemente similares a las obtenidas por el grupo de control con respecto a las métricas de la AF.

TABLA XVI
PUNTUACIONES MEDIAS DE LAS VARIABLES DE LA AF (GE VS GC)

Grupos	COMF	CORF	PERF
Experimental	0,48	0,80	0,63
Control	0,50	0,81	0,63

Fuente: esta investigación.

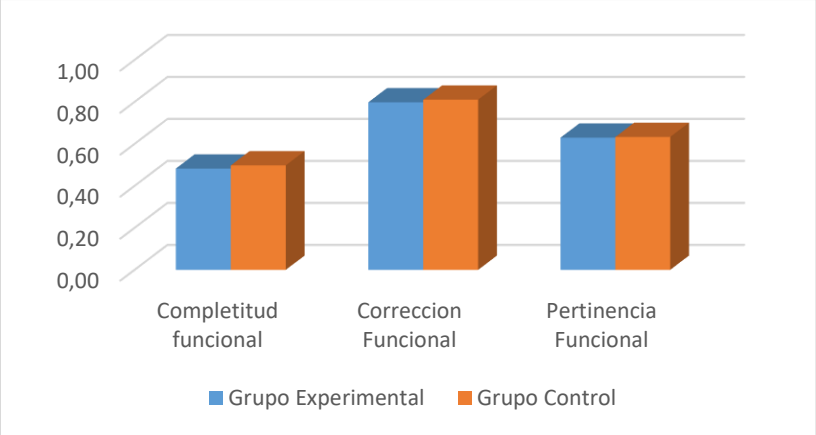


Fig. 16. Contraste de puntuaciones de las variables de la AF, grupo experimental vs grupo de control.

Fuente: esta investigación.

Inferencia estadística

Con el objetivo de brindar una conclusión estadísticamente sólida respecto al tratamiento experimental propuesto, se realizó un análisis estadístico mediante las pruebas de U de Mann Whitney y T Student, empleadas para la comparación de dos muestras independientes, buscando confirmar estadísticamente la diferencia existente entre las puntuaciones obtenidas por el grupo experimental frente a las obtenidas por el grupo de control, es decir, una diferencia general en las variables dependientes.

Las dos pruebas anteriormente mencionadas fueron aplicadas teniendo en cuenta los resultados de la prueba de normalidad, que, para este trabajo se utilizó la prueba estadística Shapiro-Wilk [175]; debido a que el tamaño de la muestra no supera los 50 datos. El nivel de confiabilidad manejado en dicha prueba fue del 95%. Los resultados se muestran en la TABLA XVII**¡Error! No se encuentra el origen de la referencia.**.

TABLA XVII
PRUEBA DE NORMALIDAD PARA EL CONJUNTO DE DATOS DE LAS VARIABLES DEPENDIENTES

Métrica	p-value
CBO	0,001
DIT	0,000
LCOM	0,314
NOC	0,000
RFC	0,010
WMC	0,001
COMF	0,053
CORF	0,077
PERF	0,600

Fuente: esta investigación.

Al realizar la prueba de normalidad de Shapiro-Wilk, si el valor de p-value obtenido es mayor que el nivel de significancia, se puede concluir que los datos siguen una distribución normal. En este caso, se puede utilizar la prueba T Student para comparar las medias de dos muestras

independientes. Por otro lado, si el valor del p-value es menor que el nivel de significancia, se concluye que los datos no siguen una distribución normal y se debe recurrir a una prueba no paramétrica, como la U de Mann Whitney. Es importante realizar estas pruebas para asegurar que se utilizan las herramientas estadísticas adecuadas y se obtienen conclusiones precisas y confiables.

Los resultados de la aplicación de las pruebas U de Mann Whitney y T Student para cada uno de los contrastes se muestran en las Tablas XVIII - XXVI, los cuales se obtuvieron mediante la herramienta SPSS™ con un nivel de confianza del 95% y teniendo en cuenta las siguientes hipótesis:

- H_0: Las medias de las puntuaciones obtenidas por los grupos en la post-prueba son similares.
- H_1: Las medias de las puntuaciones obtenidas por los grupos en la post-prueba son diferentes.

TABLA XVIII
PRUEBA U DE MANN WHITNEY PARA GHBDM VS GFPE CON RESPECTO A CBO

Variable	Tipo de Grupo		Z	U	p
	Experimental (G1) n = 10	Control (G2) n = 5			
	Rango promedio	Rango promedio			
CBO	7,50	9,00	-,613	20,000	,540

Fuente: esta investigación.

Al comparar el grupo experimental G1 con el grupo de control G2, se obtuvo un valor de p de ,540. Como este valor es mayor que 0,05, no hay evidencia estadística para rechazar la hipótesis nula (H0) ni para aceptar la hipótesis alternativa (H1), es decir, no hay evidencia suficiente con un nivel de significación del 5% de que las medias de las puntuaciones obtenidas por los estudiantes con formación homogénea y las obtenidas por aquellos con formación por su preferencia, son estadísticamente diferentes.

TABLA XIX
PRUEBA U DE MANN WHITNEY PARA GHBDM VS GFPE CON RESPECTO A DIT

Variable	Tipo de Grupo		Z	U	p
	Experimental (G1) n = 10	Control (G2) n = 5			
	Rango promedio	Rango promedio			
DIT	8,50	7,00	-1,035	20,000	,301

Fuente: esta investigación.

Al comparar el grupo experimental G1 con el grupo de control G2, se obtuvo un valor de p de ,301. Como este valor es mayor que 0,05, no hay evidencia estadística para rechazar la hipótesis nula (H0) ni para aceptar la hipótesis alternativa (H1), es decir, no hay evidencia suficiente con un nivel de significación del 5% de que las medias de las puntuaciones obtenidas por los estudiantes con formación homogénea y las obtenidas por aquellos con formación por su preferencia, son estadísticamente diferentes.

TABLA XX PRUEBA T STUDENT PARA GHBDM VS GFPE CON RESPECTO A LCOM							
Variable	Tipo de Grupo				df	t	p
	Experimental (G1) n = 10		Control (G2) n = 5				
	M	SD	M	SD			
LCOM	2,1012	,59644	2,0295	,87275	13	,189	,853

Fuente: esta investigación.

Al comparar el grupo experimental G1 con el grupo de control G2, se obtuvo un valor de p de ,853. Como este valor es mayor que 0,05, no hay evidencia estadística para rechazar la hipótesis nula (H0) ni para aceptar la hipótesis alternativa (H1), es decir, no hay evidencia suficiente con un nivel de significación del 5% de que las medias de las puntuaciones obtenidas por los estudiantes con formación homogénea y las obtenidas por aquellos con formación por su preferencia, son estadísticamente diferentes.

TABLA XXI PRUEBA U DE MANN WHITNEY PARA GHBDM VS GFPE CON RESPECTO A NOC					
Variable	Tipo de Grupo		Z	U	p
	Experimental (G1) n = 10	Control (G2) n = 5			
	Rango promedio	Rango promedio			
NOC	8,50	7,00	-1,035	20,000	,301

Fuente: esta investigación.

Al comparar el grupo experimental G1 con el grupo de control G2, se obtuvo un valor de p de ,301. Como este valor es mayor que 0,05, no hay evidencia estadística para rechazar la hipótesis nula (H0) ni para aceptar la hipótesis alternativa (H1), es decir, no hay evidencia suficiente con un nivel de significación del 5% de que las medias de las puntuaciones obtenidas por los estudiantes con formación homogénea y las obtenidas por aquellos con formación por su preferencia, son estadísticamente diferentes.

TABLA XXII PRUEBA U DE MANN WHITNEY PARA GHBDM VS GFPE CON RESPECTO A RFC					
Variable	Tipo de Grupo		Z	U	p
	Experimental (G1) n = 10	Control (G2) n = 5			
	Rango promedio	Rango promedio			
RFC	7,30	9,40	-,857	18,000	,391

Fuente: esta investigación.

Al comparar el grupo experimental G1 con el grupo de control G2, se obtuvo un valor de p de ,391. Como este valor es mayor que 0,05, no hay evidencia estadística para rechazar la hipótesis nula (H0) ni para aceptar la hipótesis alternativa (H1), es decir, no hay evidencia suficiente con un nivel

de significación del 5% de que las medias de las puntuaciones obtenidas por los estudiantes con formación homogénea y las obtenidas por aquellos con formación por su preferencia, son estadísticamente diferentes.

TABLA XXIII
PRUEBA U DE MANN WHITNEY PARA GHBDM VS GFPE CON RESPECTO A WMC

Variable	Tipo de Grupo		Z	U	p
	Experimental (G1) n = 10	Control (G2) n = 5			
	Rango promedio	Rango promedio			
WMC	8,30	7,40	-,367	22,000	,713

Fuente: esta investigación.

Al comparar el grupo experimental G1 con el grupo de control G2, se obtuvo un valor de p de ,713. Como este valor es mayor que 0,05, no hay evidencia estadística para rechazar la hipótesis nula (H0) ni para aceptar la hipótesis alternativa (H1), es decir, no hay evidencia suficiente con un nivel de significación del 5% de que las medias de las puntuaciones obtenidas por los estudiantes con formación homogénea y las obtenidas por aquellos con formación por su preferencia, son estadísticamente diferentes.

TABLA XXIV
PRUEBA T STUDENT PARA GHBDM VS GFPE CON RESPECTO A COMF

Variable	Tipo de Grupo				df	t	p
	Experimental (G1) n = 10		Control (G2) n = 8				
	M	SD	M	SD			
COMF	,4824	,12374	,4975	,19746	16	-,200	,844

Fuente: esta investigación.

Al comparar el grupo experimental G1 con el grupo de control G2, se obtuvo un valor de p de ,844. Como este valor es mayor que 0,05, no hay evidencia estadística para rechazar la hipótesis nula (H0) ni para aceptar la hipótesis alternativa (H1), es decir, no hay evidencia suficiente con un nivel de significación del 5% de que las medias de las puntuaciones obtenidas por los estudiantes con formación homogénea y las obtenidas por aquellos con formación por su preferencia, son estadísticamente diferentes.

TABLA XXV
PRUEBA T STUDENT PARA GHBDM VS GFPE CON RESPECTO A CORF

Variable	Tipo de Grupo				df	t	p
	Experimental (G1) n = 10		Control (G2) n = 8				
	M	SD	M	SD			
CORF	,7979	,16833	, 8103	,06881	16	-,195	,847

Fuente: esta investigación.

Al comparar el grupo experimental G1 con el grupo de control G2, se obtuvo un valor de p de ,847. Como este valor es mayor que 0,05, no hay evidencia estadística para rechazar la hipótesis nula

(H0) ni para aceptar la hipótesis alternativa (H1), es decir, no hay evidencia suficiente con un nivel de significación del 5% de que las medias de las puntuaciones obtenidas por los estudiantes con formación homogénea y las obtenidas por aquellos con formación por su preferencia, son estadísticamente diferentes.

TABLA XXVI
PRUEBA T STUDENT PARA GHBDM VS GFPE CON RESPECTO A PERF

Variable	Tipo de Grupo				df	t	p
	Experimental (G1) n = 10		Control (G2) n = 8				
	M	SD	M	SD			
PERF	,6294	,14654	, 6328	,13842	16	-,051	,960

Fuente: esta investigación.

Al comparar el grupo experimental G_1 con el grupo de control G_2, se obtuvo un valor de p de ,960. Como este valor es mayor que 0,05, no hay evidencia estadística para rechazar la hipótesis nula (H_0) ni para aceptar la hipótesis alternativa (H_1), es decir, no hay evidencia suficiente con un nivel de significación del 5% de que las medias de las puntuaciones obtenidas por los estudiantes con formación homogénea y las obtenidas por aquellos con formación por su preferencia, son estadísticamente diferentes.

Este conjunto de resultados permite destacar lo siguiente:

- Con base a las pruebas de contraste de hipótesis de los datos a través de las pruebas de U de Mann-Whitney y T Student, se evidencia que no existe una diferencia significativa que permita afirmar que la formación de equipos homogéneos bajo la dimensión de meticulosidad haya tenidos mejores resultados con respecto a las variables dependientes relacionadas con la mantenibilidad y la AF frente a los equipos formados por preferencia de los estudiantes.
- Este resultado permite establecer que la formación de grupos homogéneos basada en la dimensión de meticulosidad, parece no ser la estrategia más adecuada para desarrollar software que contribuya a obtener resultados superiores; al menos con las variables evaluadas para los atributos de mantenibilidad en sistemas OO y la AF, frente a los grupos formados por preferencia de los estudiantes. Es decir, sin importar como se formaron los grupos; los resultados fueron iguales, teniendo en cuenta que el experimento se llevó a cabo en entornos académicos.

Por otro lado, atendiendo a las métricas CK consideradas en la literatura, las cuales fueron definidas como las variables dependientes para la mantenibilidad, y tomando en cuenta los umbrales deseados de cada una de estas (ver TABLA IX), se puede establecer lo siguiente con base a los resultados anteriormente referenciados:

- Tanto el grupo de control como experimental obtuvieron puntuaciones medias (ver TABLA XIII) que están fuera del umbral deseado para las siguientes métricas: RFC, LCOM, CBO y WMC. A pesar de estos resultados, cabe destacar que el grupo experimental presentó una puntuación media en las métricas CBO y RFC ligeramente inferior al grupo de control. Pese a que la prueba de contraste de hipótesis concluye que no se puede hablar de una diferencia estadísticamente significativa valdría la pena entonces, refinar el experimento enfocadas en dichas métricas en búsqueda de la significancia deseada.

- El grupo experimental obtuvo un valor promedio de 3.71 para la métrica CBO, que se encuentra cercano al umbral óptimo recomendado de 1 a 3. Esto sugiere que existe una posible relación positiva para dicha métrica. Asimismo, para la métrica RFC, cuyo umbral deseado es de 5 a 10, el grupo experimental obtuvo un valor promedio ligeramente superior al umbral deseado de 12.19.
- En cuanto a las métricas DIT y NOC, el grupo de control y experimental obtuvieron valores promedio de cero, tener cero o casi ningún valor tampoco es útil [49], esto pudo haber ocurrido debido a que los grupos no aplicaron la herencia entre clases.

Con respecto a los resultados anteriormente referenciados y las puntuaciones medias obtenidas para las métricas de AF (ver TABLA XVI**¡Error! No se encuentra el origen de la referencia.**), se puede establecer lo siguiente:

- En cuanto a la variable COMF, el grupo de control obtuvo un valor promedio de 0,50 y el grupo experimental un valor promedio de 0,48. Lo que significa un puntaje bajo para ambos grupos; considerando que entre más cerca esté a 1 es más favorable. Por ende, se podría mencionar que no se cumplió con el conjunto total de funcionalidades que cubre todas las tareas y los objetivos del usuario final. Sin embargo, el grupo de control presenta una puntuación en la métrica COMF ligeramente superior al grupo experimental. Pese a que la prueba de contraste de hipótesis concluye que no se puede hablar de una diferencia estadísticamente significativa, también valdría la pena refinar el experimento enfocadas en dicha métrica en búsqueda de la significancia deseada.
- En cuanto a la variable CORF, se observó que el grupo de control obtuvo un valor promedio de 0,81 y el grupo experimental obtuvo un valor promedio de 0,80. Estos valores son considerados buenos ya que se acercan a 1, lo que indica que el producto o sistema tiene la capacidad de proporcionar resultados correctos con el nivel de precisión requerido. Es importante destacar que estos valores promedio se calcularon en base al total de funcionalidades implementadas (COMF).
- En cuanto a la variable PERF, tanto el grupo de control como experimental obtuvieron un puntaje del 0,63. Es decir, dicho valor se considera bajo al estar lejos de 1, y equivale al grado en el que las funciones del sistema facilitan la realización de las tareas y los objetivos especificados. Teniendo en cuenta que esos valores promedio corresponden al total de funcionalidades implementadas (COMF).

V. ANÁLISIS Y DISCUSIÓN DE LOS RESULTADOS

Es importante destacar que, en el grupo de control, únicamente 8 de los 9 grupos entregaron sus trabajos, y la AF del producto software fue evaluada sobre estos. De los 8 grupos evaluados, solo 5 desarrollaron el software en JAVA, mientras que los demás utilizaron otros lenguajes de programación. Esta situación limitó la evaluación de las variables dependientes relacionadas con la mantenibilidad a los 5 grupos que utilizaron JAVA. Si bien es cierto que las métricas CK son independientes del lenguaje de programación, es relevante considerar que la complejidad y la estructura del código pueden variar entre los diferentes lenguajes de programación utilizados, lo que podría afectar la interpretación y comparación de las métricas CK en los distintos proyectos.

Considerando lo anterior y a partir de los resultados presentados, se puede concluir que no se encontraron diferencias estadísticamente significativas entre el grupo experimental G1 y el grupo de control G2 en relación a las variables dependientes relacionadas con la mantenibilidad y la AF. En otras palabras, la formación de grupos homogéneos basados en la dimensión de la meticulosidad no arrojó los resultados esperados en comparación con los grupos formados por preferencia estudiantil. Como consecuencia, se aceptan las hipótesis nulas planteadas en este estudio.

Por otro lado, resulta importante comprender las puntuaciones medias de las variables dependientes en relación a la mantenibilidad obtenidas por los grupos (experimental y de control) en relación a los umbrales presentados en la TABLA IX**¡Error! No se encuentra el origen de la referencia.**. Donde se destaca que las puntuaciones medias de las métricas CBO y RFC por parte del grupo experimental, obtuvieron un valor ligeramente superior al umbral deseado y ligeramente inferior al promedio obtenido por el grupo experimental. Esto sugiere que el nivel de meticulosidad en la formación de grupos podría estar influyendo positivamente a dichas métricas, las cuales ayudan a crear un código más comprensible, modular y resistente a cambios, lo que facilita la evolución y el mantenimiento a lo largo del tiempo. En cuanto al grupo experimental, todas las puntuaciones de dichas variables se situaron por fuera del umbral recomendado.

Con respecto a las variables dependientes en relación a la AF, se observó que los valores promedio para ambos grupos (experimental y de control) fueron relativamente bajos, lo que indica que no se logró un alto grado de cumplimiento de los requisitos funcionales del software. Sin embargo, se encontró que el grupo de control obtuvo puntuaciones medias en las métricas CORF y COMF, ligeramente superiores al grupo experimental. Esto sugiere que los grupos formados por preferencia de los estudiantes podrían tener una mayor habilidad para comprender y cumplir con los requerimientos funcionales del software que están desarrollando, aunque no alcanzaron resultados satisfactorios en estas variables.

Es probable que una muestra más amplia hubiera permitido obtener resultados más concluyentes, lo que podría explicar la falta de relaciones estadísticamente significativas entre las variables dependientes evaluadas. Es comprensible que los resultados obtenidos no pueden considerarse concluyentes ni fácilmente generalizables a la población completa de ingenieros de software en la industria. La elección de estudiantes como sujetos de estudio, aunque permitió un enfoque controlado y accesible para la investigación, presenta limitaciones inherentes en términos de representatividad y experiencia práctica en el campo de la IS. De hecho, a la luz de esto se subraya la necesidad de cautela al interpretar los resultados y considerarlos en un contexto más específico. Por ello, este estudio no pretende ser un reflejo completo de la diversidad de experiencias y competencias presentes en el mundo real de la IS, y es fundamental que el lector sea consciente de las limitaciones al considerar las implicaciones de los resultados.

Aunque son escasos los estudios que aborden los efectos de la personalidad en los atributos de calidad del software [21], resulta valioso contextualizar los hallazgos de este estudio a través de comparaciones con investigaciones previas. Por ejemplo, S. Romano et al. [16] encontraron una relación positiva entre la meticulosidad, la extroversión, el neuroticismo y la calidad del software, especialmente en términos de las métricas de McCabe y Halstead. Sin embargo, en el presente estudio no se observó una correlación positiva entre la meticulosidad y las métricas CK y de adecuación funcional. Estas discrepancias pueden explicarse por diferencias metodológicas, ya que el presente estudio se enfocó en desarrollo de software en equipos de trabajo y evaluó solo una dimensión, en contraste con el estudio previo, que se enfocó en evaluaciones individuales y consideró todas las dimensiones del modelo Big Five. Además, es importante destacar que las métricas de calidad del software pueden variar ampliamente según el enfoque y los objetivos de cada estudio.

Similarmente, Barroso et al. [58] estudiaron la relación individual entre la personalidad de los desarrolladores y la calidad del software, a través de métricas de software OO, encontrando que la meticulosidad no se correlacionó con ninguna métrica, pero sí la dimensión de neuroticismo, la cual mostró una relación positiva: a mayor puntuación en esta dimensión, menor probabilidad de experimentar síntomas de depresión y mayor estabilidad emocional. Estas cualidades pueden mejorar la calidad del software, especialmente en métricas como CC, CBO y DIT, aunque el presente estudio coincide en que la meticulosidad no tiene un vínculo positivo con métricas CK. Se respalda así la recomendación de realizar más experimentos, ampliando la muestra y variando el entorno de desarrollo, como la ubicación geográfica, para obtener una comprensión más completa de esta relación.

Por otro lado, Acuña et al. [36] encontraron una relación positiva entre la extroversión y la calidad del software en proyectos ágiles, donde los grupos de trabajo se formaron al azar. El producto de software fue evaluado mediante una fórmula general que incluyó la descomposición y modularización, la capacidad de prueba, la adecuación funcional, la reutilización, el estilo de programación y la participación de los miembros del equipo. De igual forma, se evidenció que la meticulosidad no influye en la calidad del software, y aunque ambos estudios abordan aspectos relativos a ello, el presente se distingue por explorar un conjunto más amplio de métricas.

Además, en cuanto al tipo de formación de los grupos, se puede notar diferentes puntos de vista entre los hallazgos de este estudio y los resultados de otros trabajos relacionados. El presente estudio no obtuvo buenos resultados en cuanto a la formación de grupos homogéneos bajo la dimensión de la meticulosidad, mientras que Pieterse et al. [39] encontraron que los grupos heterogéneos tuvieron un mejor desempeño que los homogéneos, especialmente durante las primeras fases de crecimiento del equipo, Choi et al. [64] concluyeron que las parejas con personalidades heterogéneas fueron más efectivas que las parejas con personalidades similares en programación en parejas. De manera similar, Sfetsos et al. [63] encontraron que las parejas con personalidades y temperamentos heterogéneos tuvieron una mejor comunicación, desempeño y viabilidad de colaboración. Además, Peeters et al. [38] descubrieron que los equipos homogéneos tuvieron un mejor rendimiento para resolver tareas estructuradas, mientras que los equipos heterogéneos fueron más útiles para resolver tareas no estructuradas. A pesar de estas diferencias, Sánchez et al. [28] destacaron la importancia de la formación de grupos homogéneos en escenarios de aprendizaje colaborativo y sugirieron que los rasgos de personalidad pueden ser un criterio efectivo para la formación de grupos en este contexto.

En resumen, estos hallazgos resaltan la importancia de continuar explorando esta área, la cual ha sido objeto de gran interés en la IS [20]–[22], especialmente dado que los rasgos de personalidad han sido identificados como un factor crucial en el desarrollo de software [13], [43]. No todos los individuos pueden destacarse en todas las tareas, por lo que asignar a cada persona a sus tareas preferidas en el proyecto podría aumentar las posibilidades de éxito en el desarrollo de software [176].

VI. CONCLUSIONES

La formación de grupos homogéneos en términos de la dimensión de meticulosidad no parece tener un impacto significativo en la calidad del software, específicamente en lo que respecta a la mantenibilidad y la adecuación funcional. No se observaron ventajas o resultados superiores en comparación con los equipos formados por preferencia de los estudiantes. Estos hallazgos sugieren que la meticulosidad en sí misma no garantiza necesariamente una mejora significativa en la calidad del software en el contexto estudiado. Aunque las limitaciones, como el tamaño de la muestra, experticia de los estudiantes y el tiempo para desarrollar el software pudieron haber influido en los resultados, este estudio contribuye un valioso punto de partida para la exploración de la relación entre las diferentes dimensiones de la personalidad y los atributos de calidad del software en contextos de trabajo en grupo.

Este estudio subraya la importancia de continuar explorando las dimensiones de la personalidad que pueden aportar positivamente al desarrollo de software de calidad. A pesar de que los resultados del presente estudio no han revelado diferencias significativas, no se puede subestimar la importancia de la meticulosidad en el desarrollo de software, en lugar de considerarla de manera aislada, investigaciones futuras podrían centrarse en la formación de grupos mixtos, donde se integren otras dimensiones de personalidad como la extroversión y el neuroticismo, que han cobrado relevancia en otros estudios [14], [33], [50]. La combinación de estas dimensiones podría dar lugar a equipos altamente potenciales que contribuyan significativamente a la creación de software de calidad. En este sentido, la búsqueda de la combinación óptima de rasgos de personalidad sigue siendo un área emocionante y valiosa para la investigación en la IS.

Por otro lado, es importante destacar que los resultados presentados en este estudio no solo aportan conocimiento valioso al campo de la IS, sino también señalan la complejidad inherente a la relación entre la personalidad y la calidad del software [18]. Esto resalta la necesidad de considerar cuidadosamente los contextos y enfoques metodológicos al explorar esta dinámica. Es esencial destacar que el uso de pruebas de personalidad adecuadas y validadas es fundamental para llevar a cabo un análisis preciso de la influencia de la personalidad en la calidad del software. Además, estos estudios se beneficiarían enormemente al trabajar en conjunto otras disciplinas como la psicología y la sociología. Esta colaboración interdisciplinaria facilitará una evaluación más completa de la personalidad y enriquecerá la investigación en el campo de la IS, lo que puede resultar fundamental para descubrir nuevas perspectivas en este tipo de estudios.

Finalmente, y no menos importante, se debe atender al llamado que se hace en la literatura en indagar esta área que tiene mucha importancia en la IS, mucho campo por explorar y la baja existencia de estudios relacionados entre los rasgos de personalidad y los atributos de calidad de software [21]–[23], [58].

VII. RECOMENDACIONES

Este estudio es solo un primer paso hacia la comprensión de la relación entre la personalidad y los atributos calidad del software. Se sugiere seguir explorando qué dimensiones específicas de la personalidad pueden influir en los diferentes atributos de calidad del software y en general de la ciencia de la computación. A partir de esto, eventualmente proponer cual sería la formación adecuada que afecte de forma positiva a lo anteriormente mencionado.

Así mismo, se sugiere repetir en nuevas ocasiones el experimento descrito, en una de esas ocasiones es formar equipos heterogéneos o mixtos. Además de tener en cuenta la dimensión de meticulosidad, incluir otras dimensiones como extroversión y amabilidad, ya que estudios anteriores afirmaron una influencia positiva en el desempeño y en las métricas de calidad del software [23], [36], [92].

Por otro lado, se sugiere ampliar la muestra de participantes para obtener resultados más concluyentes, ya que esto podría permitir establecer relaciones estadísticamente significativas entre las variables evaluadas. Además, se sugiere refinar el experimento con enfoque en las métricas CBO y RFC, debido a que se encontró una ligera influencia del nivel de meticulosidad en la formación de grupos en estas métricas.

VIII. TRABAJOS FUTUROS

La personalidad desempeña un papel crucial en la IS, y se vislumbran diversas áreas de investigación y trabajos futuros que podrían profundizar en la comprensión de la relación entre la personalidad y la calidad del software. Una sugerencia para investigaciones futuras sería la realización de experimentos similares en la industria del software, lo que proporcionaría resultados más aplicables y precisos en entornos reales. Para ello, se podría considerar la participación de programadores profesionales con niveles equivalentes de experiencia, permitiendo evaluar la relación entre la personalidad y los atributos de calidad del software en un contexto más próximo a la realidad. Además, se deberían explorar otros factores como la influencia del género y la edad, ya que estos pueden tener un impacto significativo en el desempeño y los resultados relacionados con la calidad del software. Un área de investigación adicional podría centrarse en la relación entre la personalidad y la creatividad en el desarrollo de software, como un área clave debido a su potencial impacto en la calidad del software. Comprender cómo los distintos rasgos de personalidad pueden influir en la capacidad creativa de los profesionales de software podría proporcionar conocimientos cruciales para potenciar la innovación y, por ende, mejorar la calidad de los productos y procesos en el ámbito de la IS. En resumen, la personalidad emerge como un tema fascinante y relevante en la IS, con numerosas áreas por explorar en futuros estudios.

REFERENCIAS

[1] H. KRASNER, "The Cost of Poor Software Quality in the US: A 2020 Report," 2020. [Online]. Available: https://www.it-cisq.org/cisq-files/pdf/CPSQ-2020-report.pdf.

[2] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education, 2015.

[3] L. Machuca-Villegas, G. P. Gasca-Hurtado, S. M. Puente, and L. M. R. Tamayo, "Perceptions of the human and social factors that influence the productivity of software development teams in Colombia: A statistical analysis," *J. Syst. Softw.*, vol. 192, 2022, doi: 10.1016/j.jss.2022.111408.

[4] L. Li, J. Cao, and D. Lo, "Sentiment analysis over collaborative relationships in open source software projects," in *Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE*, 2020, vol. PartF16244, pp. 418 – 423, doi: 10.18293/SEKE2020-030.

[5] M. Sánchez-Gordón, *Connecting the dots between human factors and software engineering*. 2020.

[6] M. John, F. Maurer, and B. Tessem, "Human and social factors of software engineering," 2005, p. 686, doi: 10.1145/1062455.1062612.

[7] D. Varona, L. F. Capretz, Y. Piñero, and A. Raza, "Evolution of Software Engineers' Personality Profile," *SIGSOFT Softw. Eng. Notes*, vol. 37, no. 1, pp. 1–5, 2012, doi: 10.1145/2088883.2088901.

[8] R. M. Ryckman, *Theories of personality (8th ed.)*. Wadsworth/Thomson, 2004.

[9] M. Caulo, R. Francese, G. Scanniello, and G. Tortora, "Relationships between personality traits and productivity in a multi-platform development context," in *ACM International Conference Proceeding Series*, 2021, pp. 70 – 79, doi: 10.1145/3463274.3463327.

[10] S. E. Pek and J. H. L. Koh, "Team Formation using Character-based Gamification: Effects on Online Teamwork Experience During COVID-19," in *2021 16th International Conference on Computer Science & Education (ICCSE)*, 2021, pp. 247–252, doi: 10.1109/ICCSE51940.2021.9569643.

[11] Z. Huang *et al.*, "Predicting Community Smells' Occurrence on Individual Developers by Sentiments," in *2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC)*, 2021, pp. 230–241, doi: 10.1109/ICPC52881.2021.00030.

[12] J. Agarwal, E. Piatt, and P. K. Imbrie, "Team formation in engineering classrooms using multi-criteria optimization with genetic algorithms," in *2022 IEEE Frontiers in Education Conference (FIE)*, 2022, pp. 1–6, doi: 10.1109/FIE56618.2022.9962741.

[13] E. Weilemann, "A Winning Team - What Personality Has To Do With Software Engineering," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, May 2019, pp. 252–253, doi: 10.1109/ICSE-Companion.2019.00100.

[14] N. Qamar and A. A. Malik, "Birds of a Feather Gel Together: Impact of Team Homogeneity

on Software Quality and Team Productivity," *IEEE Access*, vol. 7, pp. 96827–96840, 2019, doi: 10.1109/ACCESS.2019.2929152.

[15] M. Iqbal, F. Ammar, A. Aldaihani, T. Khan, and A. Shah, "Predicting Most Effective Software Development Teams by Mapping MBTI Personality Traits with Software Lifecycle Activities," 2019, pp. 1–5, doi: 10.1109/ICETAS48360.2019.9117370.

[16] S. Romano, G. Scanniello, and P. Dionisio, "On the Role of Personality Traits in Implementation Tasks: A Preliminary Investigation with Students," in *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2022, pp. 189–196, doi: 10.1109/SEAA56994.2022.00037.

[17] M. Chugh, A. Pandey, and S. Vyas, "A Comprehensive Study on the Association Between Personality Traits and Software Development," 2022, doi: 10.1145/3590837.3590900.

[18] R. Gilal, M. Omar, and M. M. Rejab, "INVESTIGATING THE RELATIONSHIP OF PERSONALITY TYPES AND TIME PRESSURE AMONG SOFTWARE DEVELOPERS BASED ON A RULE-BASED APPROACH," *ARPN J. Eng. Appl. Sci.*, vol. 17, no. 19, pp. 1811 – 1819, 2022, [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85164829613&partnerID=40&md5=d763413a424b50c597a345fcbfb06a30.

[19] D. Hidellaarachchi, J. Grundy, R. Hoda, and K. Madampe, "The Effects of Human Aspects on the Requirements Engineering Process: A Systematic Literature Review," *IEEE Trans. Softw. Eng.*, vol. 48, no. 6, pp. 2105 – 2127, 2022, doi: 10.1109/TSE.2021.3051898.

[20] S. Cruz, F. Q. B. da Silva, and L. F. Capretz, "Forty years of research on personality in software engineering: A mapping study," *Comput. Human Behav.*, vol. 46, pp. 94–113, 2015, doi: https://doi.org/10.1016/j.chb.2014.12.008.

[21] E. Weilemann and P. Brune, "The Influence of Personality on Software Quality – A Systematic Literature Review," in *Advances in Intelligent Systems and Computing*, vol. 1159 AISC, 2020, pp. 766–777.

[22] E. Guveyi, M. S. Aktas, and O. Kalipsiz, "Human Factor on Software Quality: A Systematic Literature Review," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2020, vol. 12252 LNCS, pp. 918–930, doi: 10.1007/978-3-030-58811-3_65.

[23] A. S. Barroso, K. H. de J. Prado, M. S. Soares, and R. P. C. do Nascimento, "How Personality Traits Influences Quality of Software Developed by Students," 2019, doi: 10.1145/3330204.3330237.

[24] N. M. Seel, Ed., "Big Five Personality: Five-Factor Personality Theory," in *Encyclopedia of the Sciences of Learning*, Boston, MA: Springer US, 2012, p. 454.

[25] S. L. Kichuk and W. H. Wiesner, "The big five personality factors and team performance: implications for selecting successful product design teams," *J. Eng. Technol. Manag.*, vol. 14, no. 3, pp. 195–221, 1997, doi: https://doi.org/10.1016/S0923-4748(97)00010-6.

[26] M. V. Kosti, R. Feldt, and L. Angelis, "Archetypal Personalities of Software Engineers and Their Work Preferences: A New Perspective for Empirical Studies," *Empir. Softw. Engg.*, vol. 21, no. 4, pp. 1509–1532, 2016, doi: 10.1007/s10664-015-9395-3.

[27] E. Weilemann and P. Brune, “How to Staff Software Engineering Team Roles Using the Concept of Personality? – An Exploratory Study,” *Adv. Intell. Syst. Comput.*, vol. 1367 AISC, pp. 271 – 284, 2021, doi: 10.1007/978-3-030-72660-7_26.

[28] O. R. Sánchez, C. A. Collazos Ordóñez, M. Á. Redondo Duque, and I. Ibert Bittencourt Santana Pinto, “Homogeneous Group Formation in Collaborative Learning Scenarios: An Approach Based on Personality Traits and Genetic Algorithms,” *IEEE Trans. Learn. Technol.*, vol. 14, no. 4, pp. 486–499, 2021, doi: 10.1109/TLT.2021.3105008.

[29] S. Wagner and M. Ruhe, “A Systematic Review of Productivity Factors in Software Development,” *CoRR*, vol. abs/1801.0, 2018, [Online]. Available: http://arxiv.org/abs/1801.06475.

[30] S. L. Pfleeger, *Software engineering : theory and practice*, 4th ed. Upper Saddle River [N.J: Prentice Hall, 2010.

[31] M. Rehman, S. Safdar, A. K. Mahmood, A. Amin, and R. Salleh, “Personality traits and knowledge sharing behavior of software engineers,” in *2017 8th International Conference on Information Technology (ICIT)*, May 2017, pp. 6–11, doi: 10.1109/ICITECH.2017.8079908.

[32] G. Matturro, F. Raschetti, and C. Fontán, “A Systematic Mapping Study on Soft Skills in Software Engineering.”

[33] M. Yilmaz, R. V O’Connor, R. Colomo-Palacios, and P. Clarke, “An examination of personality traits and how they impact on software development teams,” *Inf. Softw. Technol.*, vol. 86, pp. 101–122, 2017, doi: https://doi.org/10.1016/j.infsof.2017.01.005.

[34] A. Amin *et al.*, “The impact of personality traits and knowledge collection behavior on programmer creativity,” *Inf. Softw. Technol.*, vol. 128, p. 106405, 2020, doi: https://doi.org/10.1016/j.infsof.2020.106405.

[35] T. Walle and J. E. Hannay, “Personality and the nature of collaboration in pair programming,” in *2009 3rd International Symposium on Empirical Software Engineering and Measurement*, 2009, pp. 203–213, doi: 10.1109/ESEM.2009.5315996.

[36] S. T. Acuña, M. N. Gómez, J. E. Hannay, N. Juristo, and D. Pfahl, “Are team personality and climate related to satisfaction and software quality? Aggregating results from a twice replicated experiment,” *Inf. Softw. Technol.*, vol. 57, no. 1, pp. 141–156, 2015, doi: 10.1016/j.infsof.2014.09.002.

[37] J. Gulati, P. Bhardwaj, B. Suri, and A. Lather, “A Study of Relationship between Performance, Temperament and Personality of a Software Programmer,” *ACM SIGSOFT Softw. Eng. Notes*, vol. 41, pp. 1–5, 2016, doi: 10.1145/2853073.2853089.

[38] M. A. G. Peeters, H. F. J. M. van Tuijl, C. G. Rutte, and I. M. M. J. Reymen, “Personality and team performance: a meta-analysis,” *Eur. J. Pers.*, vol. 20, no. 5, pp. 377–396, 2006, doi: 10.1002/per.588.

[39] V. Pieterse, D. G. Kourie, and I. P. Sonnekus, “Software Engineering Team Diversity and Performance,” in *Proceedings of the 2006 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in*

Developing Countries, 2006, pp. 180–186, doi: 10.1145/1216262.1216282.

[40] J. Karn and T. Cowling, "A Follow up Study of the Effect of Personality on the Performance of Software Engineering Teams," in *Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering*, 2006, pp. 232–241, doi: 10.1145/1159733.1159769.

[41] A. Gilal, J. Jaafar, A. Abro, M. Omar, S. Basri, and M. Q. Saleem, "Effective Personality Preferences of Software Programmer: A Systematic Review," *J. Inf. Sci. Eng.*, vol. 33, pp. 1399–1416, 2017.

[42] M. Ahmad, W. H. Butt, and A. Ahmad, "Advance Recommendation System for the Formation of More Prolific and Dynamic Software Project Teams," in *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*, 2019, pp. 161–165, doi: 10.1109/ICSESS47205.2019.9040791.

[43] L. F. Capretz and F. Ahmed, "Why Do We Need Personality Diversity in Software Engineering?," *SIGSOFT Softw. Eng. Notes*, vol. 35, no. 2, pp. 1–11, 2010, doi: 10.1145/1734103.1734111.

[44] L. F. Capretz, F. Ahmed, and F. Q. B. da Silva, "Soft sides of software," *arXiv Prepr. arXiv1711.07876*, 2017, doi: https://doi.org/10.1016/j.infsof.2017.07.011.

[45] M. Wiesche and H. Krcmar, "The relationship of personality models and development tasks in software engineering," *SIGMIS-CPR 2014 - Proc. 2014 Conf. Comput. People Res.*, 2014, doi: 10.1145/2599990.2600012.

[46] S. S. J. O. Cruz, F. Q. B. da Silva, C. V. F. Monteiro, P. Santos, I. Rossilei, and M. T. dos Santos, "Personality in software engineering: Preliminary findings from a systematic literature review," in *15th Annual Conference on Evaluation Assessment in Software Engineering (EASE 2011)*, 2011, pp. 1–10, doi: 10.1049/ic.2011.0001.

[47] A. S. Sodiya, H. Longe, S. Onashoga, A. Oludele, and L. Omotosho, "An Improved Assessment of Personality Traits in Software Engineering," 2007, doi: 10.28945/3164.

[48] J. D. Delgado Jojoa, O. Revelo Sánchez, and S. V. Chamorro, "Psychological Models and Instruments Employed to Identify Personality Traits of Software Developers: A Systematic Mapping Study," in *Human-Computer Interaction*, 2022, pp. 146–161.

[49] H. P. and D. V. K. S. V. Rai, A. M. Srivastava, "Estimation of Maintainability in Object Oriented Design Phase: State of the art," 2015.

[50] C. Monaghan, B. Bizumic, K. Reynolds, M. Smithson, L. Johns-Boast, and D. Rooy, "Performance in software development teams: The influence of personality and identifying as a group member," *Eur. J. Eng. Educ.*, 2014, doi: 10.1080/03043797.2014.914156.

[51] A. Endriulaitienė and L. Cirtautienė, "TEAM EFFECTIVENESS IN SOFTWARE DEVELOPMENT: THE ROLE OF PERSONALITY AND WORK FACTORS," *Bus. Theory Pract.*, vol. 22, pp. 55–68, 2021, doi: 10.3846/btp.2021.12824.

[52] "IEEE Standard Glossary of Software Engineering Terminology," *IEEE Std 610.12-1990*, pp. 1–84, 1990, doi: 10.1109/IEEESTD.1990.101064.

[53] A.-J. Molnar and S. Motogna, “A Study of Maintainability in Evolving Open-Source Software,” *Commun. Comput. Inf. Sci.*, vol. 1375, pp. 261 – 282, 2021, doi: 10.1007/978-3-030-70006-5_11.

[54] B. Y. Hernández Jarvio, P. Velasco-Elizondo, and E. Benitez-Guerrero, “Evaluando Adecuación Funcional y Usabilidad en Herramientas de Composición desde la Perspectiva del Usuario Final,” *RISTI - Rev. Iber. Sist. e Tecnol. Inf.*, pp. 96–114, 2016, doi: 10.17013/risti.17.96-114.

[55] A. D. Toro and B. B. Jiménez, “Metodología para la Elicitación de Requisitos de Sistemas Software,” Sevilla, 2000. [Online]. Available: http://www.lsi.us.es/docs/informes/lsi-2000-10.pdf.

[56] G. Kotonya and I. Sommerville, *Requirements Engineering: Processes and Techniques*. Wiley, 1998.

[57] S. Raghavan, G. Zelesnik, G. A. Ford, and C.-M. U. P. P. A. S. E. INST., *Lecture Notes on Requirements Elicitation*. Carnegie Mellon University, Software Engineering Institute, 1994.

[58] A. Barroso, J. Madureira, T. Souza, B. Cezario, M. Soares, and R. Nascimento, *Relationship between Personality Traits and Software Quality - Big Five Model vs. Object-oriented Software Metrics*. 2017.

[59] S. T. Acuña, M. Gómez, and N. Juristo, “How do personality, team processes and task characteristics relate to job satisfaction and software quality?,” *Inf. Softw. Technol.*, vol. 51, no. 3, pp. 627–639, 2009, doi: https://doi.org/10.1016/j.infsof.2008.08.006.

[60] N. Salleh, E. Mendes, J. Grundy, and G. S. J. Burch, “An empirical study of the effects of conscientiousness in pair programming using the five-factor personality model,” in *2010 ACM/IEEE 32nd International Conference on Software Engineering*, 2010, vol. 1, pp. 577–586, doi: 10.1145/1806799.1806883.

[61] J. Hannay, E. Arisholm, H. Engvik, and D. Sjøberg, “The Effects of Personality on Pair Programming,” *Softw. Eng. IEEE Trans.*, vol. 36, pp. 61–80, 2010, doi: 10.1109/TSE.2009.41.

[62] N. Salleh, E. Mendes, and J. Grundy, “Investigating the effects of personality traits on pair programming in a higher education setting through a family of experiments,” *Empir. Softw. Eng.*, vol. 19, 2012, doi: 10.1007/s10664-012-9238-4.

[63] P. Sfetsos, I. Stamelos, L. Angelis, and I. Deligiannis, “An Experimental Investigation of Personality Types Impact on Pair Effectiveness in Pair Programming,” *Empir. Softw. Engg.*, vol. 14, no. 2, pp. 187–226, 2009, doi: 10.1007/s10664-008-9093-5.

[64] K. S. Choi, F. P. Deek, and I. Im, “Exploring the underlying aspects of pair programming: The impact of personality,” *Inf. Softw. Technol.*, vol. 50, no. 11, pp. 1114–1126, 2008, doi: https://doi.org/10.1016/j.infsof.2007.11.002.

[65] J. S. Karn, S. Syed-Abdullah, A. J. Cowling, and M. Holcombe, “A study into the effects of personality type and methodology on cohesion in software engineering teams,” *Behav. Inf. Technol.*, vol. 26, no. 2, pp. 99 – 111, 2007, doi: 10.1080/01449290500102110.

[66] M. Shameem, C. Kumar, and B. Chandra, “A proposed framework for effective software team performance: A mapping study between the team members’ personality and team climate,” in *2017 International Conference on Computing, Communication and Automation (ICCCA)*, 2017, pp. 912–917, doi: 10.1109/CCAA.2017.8229936.

[67] R. R. McCrae and P. T. Costa Jr., “The five-factor theory of personality.,” in *Handbook of personality: Theory and research, 3rd ed.*, New York, NY, US: The Guilford Press, 2008, pp. 159–181.

[68] M. M., J. P., and C. G., “Teorías de la personalidad. Un análisis histórico del concepto y su medición,” *Psychol. Av. la Discip.*, vol. 3, pp. 81–107, 2009, [Online]. Available: https://www.redalyc.org/articulo.oa?id=297225531007.

[69] I. V Delgado, F. Vidales, and I. Leal, *Psicología general*. Limusa, 2004.

[70] G. W. Allport, *La personalidad: su configuración y desarrollo*. Herder, 1975.

[71] J. M. L., “Personalidad: esbozo de una teoría integradora,” *Psicothema*, vol. 14, pp. 693–701, 2002, [Online]. Available: https://www.redalyc.org/articulo.oa?id=72714402.

[72] S. C. Cloninger, *Teorías de la personalidad*. Pearson Educación, 2002.

[73] R. B. Cattell, M. T. Russell, A. K. S. Cattell, D. L. Karol, H. E. P. Cattell, and N. S. Cubero, *16PF-5, Cuestionario factorial de personalidad:* TEA Ediciones, S.A., 2011.

[74] V. Schmidt *et al.*, “Modelo Psicobiológico de Personalidad de Eysenck: una historia proyectada hacia el futuro,” *Rev. Int. Psicol.*, vol. 11, no. 02, pp. 1–21, 2010, doi: 10.33670/18181023.v11i02.63.

[75] A. Muñoz, “Teorías y Evaluación de la Personalidad.” Publicaciones Didácticas, pp. 86, 58–63, 2017, [Online]. Available: http://publicacionesdidacticas.com/hemeroteca/articulo/086006/articulo-pdf.

[76] S. McDonald and H. M. Edwards, “Who should test whom?,” *Commun. ACM*, vol. 50, pp. 66–71, 2007.

[77] C. G. Jung, *Tipos Psicologicos*. Edhasa, 1971.

[78] D. Keirsey, *Please Understand Me II: Temperament, Character, Intelligence*. Prometheus Nemesis, 1998.

[79] P. Costa and R. Mccrae, “Neo PI-R professional manual,” *Psychol. Assess. Resour.*, vol. 396, 1992.

[80] I. B. Myers, *MBTI Manual: A Guide to the Development and Use of the Myers-Briggs Type Indicator*. Consulting Psychologists Press, 1998.

[81] M. K. Mount and M. R. Barrick, “Five reasons why the ‘Big Five’ article has been frequently cited.,” *Pers. Psychol.*, vol. 51, pp. 849–857, 1998, doi: 10.1111/j.1744-6570.1998.tb00743.x.

[82] R. A. Aguilar, A. de Antonio, and R. Imbert, “Searching Pancho’s Soul: An Intelligent Virtual Agent for Human Teams,” in *Electronics, Robotics and Automotive Mechanics*

Conference (CERMA 2007), 2007, pp. 568–571, doi: 10.1109/CERMA.2007.4367747.

[83] O. P. John and S. Srivastava, "The Big Five Trait taxonomy: History, measurement, and theoretical perspectives.," in *Handbook of personality: Theory and research, 2nd ed.*, New York, NY, US: Guilford Press, 1999, pp. 102–138.

[84] R. L. John, O. P., Donahue, E. M., & Kentle, "The Big Five Inventory - Versions 4a and 54." 1991.

[85] G. V. Caprara, C. Barbaranelli, L. Borgogni, and M. Perugini, "The 'big five questionnaire': A new questionnaire to assess the five factor model," *Pers. Individ. Dif.*, vol. 15, no. 3, pp. 281–288, 1993, doi: https://doi.org/10.1016/0191-8869(93)90218-R.

[86] C. Barbaranelli, G. V. Caprara, A. Rabasca, and C. Pastorelli, "A questionnaire for measuring the Big Five in late childhood," *Pers. Individ. Dif.*, vol. 34, no. 4, pp. 645–664, 2003, doi: https://doi.org/10.1016/S0191-8869(02)00051-X.

[87] R. Brown, "Social Identity Theory: Past achievements, current problems and future challenges.," *Eur. J. Soc. Psychol.*, vol. 30, pp. 745–778, 2000, doi: 10.1002/1099-0992(200011/12)30:6<745::AID-EJSP24>3.0.CO;2-O.

[88] J. C. Turner, M. A. Hogg, P. J. Oakes, S. D. Reicher, and M. S. Wetherell, *Rediscovering the social group: A self-categorization theory*. Cambridge, MA, US: Basil Blackwell, 1987.

[89] R. N. Turner, R. J. Crisp, and E. Lambert, "Imagining intergroup contact can improve intergroup attitudes.," *Gr. Process. Intergr. Relations*, vol. 10, pp. 427–441, 2007, doi: 10.1177/1368430207081533.

[90] K. Whittingham, "Impact of Personality on Academic Performance of MBA Students: Qualitative Versus Quantitative Courses*," *Decis. Sci. J. Innov. Educ.*, vol. 4, pp. 175–190, 2006, doi: 10.1111/j.1540-4609.2006.00112.x.

[91] J. H. Bradley and F. J. Hebert, "The effect of personality type on team performance," *J. Manag. Dev.*, vol. 16, pp. 337–353, 1997.

[92] A. Barroso, J. Madureira, M. Soares, and R. Nascimento, *Influence of Human Personality in Software Engineering - A Systematic Literature Review*. 2017.

[93] L. Fernández-Sanz and S. Misra, "Influence of Human Factors in Software Quality and Productivity," in *Computational Science and Its Applications - ICCSA 2011*, 2011, pp. 257–269.

[94] D. R. Forsyth, *Group Dynamics, 6th ed.* Cengage Learning.

[95] D. W. Johnson, R. T. Johnson, and K. A. Smith, "Cooperative learning : increasing college faculty instructional productivity," 1991.

[96] J. C. Bean and M. Weimer, *Engaging Ideas: The Professor's Guide to Integrating Writing, Critical Thinking, and Active Learning in the Classroom*. Wiley, 2011.

[97] M. R. Barrick, G. L. Stewart, M. J. Neubert, and M. K. Mount, "Relating member ability and personality to work-team processes and team effectiveness.," *J. Appl. Psychol.*, vol. 83,

pp. 377–391, 1998.

[98] E. Aronson and et al, *The jigsaw classroom*. Oxford, England: Sage, 1978.

[99] P. Cranton, "Types of group learning," *New Dir. Adult Contin. Educ.*, vol. 1996, pp. 25–32, 2006, doi: 10.1002/ace.36719967105.

[100] Y. Sharan and S. Sharan, *Expanding Cooperative Learning Through Group Investigation*. 1992.

[101] K. W. Phillips, *Diversity and Groups*. Emerald Group Publishing Limited, 2008.

[102] A. Tziner, "How Team Composition Affects Task Performance: Some Theoretical Insights," *Psychol. Rep.*, vol. 57, no. 3_suppl, pp. 1111–1119, 1985, doi: 10.2466/pr0.1985.57.3f.1111.

[103] J. M. George, "Personality, affect, and behavior in groups.," *J. Appl. Psychol.*, vol. 75, no. 2, pp. 107–116, 1990, doi: 10.1037/0021-9010.75.2.107.

[104] M. A. Hogg, "The Social Psychology of Group Cohesiveness: From Attraction to Social Identity," 1992.

[105] S. D. Brookfield and S. Preskill, *Discussion as a Way of Teaching: Tools and Techniques for Democratic Classrooms*. Jossey-Bass; 2nd edition, 2005.

[106] N. Kerr and S. Tindale, "Group Performance and Decision Making," *Annu. Rev. Psychol.*, vol. 55, pp. 623–655, 2004, doi: 10.1146/annurev.psych.55.090902.142009.

[107] L. Sanz-Martínez, E. Er, A. Martínez-Monés, Y. Dimitriadis, and M. L. Bote-Lorenzo, "Creating collaborative groups in a MOOC: a homogeneous engagement grouping approach," *Behav. \& Inf. Technol.*, vol. 38, no. 11, pp. 1107–1121, 2019, doi: 10.1080/0144929X.2019.1571109.

[108] W. E. Beane and E. A. Lemke, "Group variables influencing the transfer of conceptual behavior.," *J. Educ. Psychol.*, vol. 62, pp. 215–218, 1971, doi: 10.1037/h0031145.

[109] I. Sommerville, *Ingeniería del software, Séptima edición Ed. Cap. 27*. Madrid: Pearson Educacion S.A, 2005.

[110] N. S. Godbole, *Software Quality Assurance: Principles and Practice*. Alpha Science International Limited, 2004.

[111] P. Nistala, K. V. Nori, and R. Reddy, "Software Quality Models: A Systematic Mapping Study," in *2019 IEEE/ACM International Conference on Software and System Processes (ICSSP)*, 2019, pp. 125–134, doi: 10.1109/ICSSP.2019.00025.

[112] P. Bourque, R. E. Fairley, and I. C. Society, *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0*, 3rd ed. Washington, DC, USA: IEEE Computer Society Press, 2014.

[113] F. N. Colakoglu, A. Yazici, and A. Mishra, "Software Product Quality Metrics: A Systematic Mapping Study," *IEEE Access*, vol. 9, pp. 44647–44670, 2021, doi: 10.1109/ACCESS.2021.3054730.

[114] C. P. Team, “CMMI for Development, Version 1.3,” Pittsburgh, PA, 2010. [Online]. Available: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9661.

[115] D. M. Ahern, A. Clouse, and R. Turner, *CMMI Distilled: A Practical Introduction to Integrated Process Improvement*. Addison-Wesley, 2004.

[116] K. Roebuck, *ISO/IEC 15504 (SPICE): High-impact Strategies - What You Need to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors*. TEBBO/Emero Publishing, 2011.

[117] S. T. Acuna, N. Juristo, A. M. Moreno, and A. Mon, *A Software Process Model Handbook for Incorporating People’s Capabilities*. Springer US, 2006.

[118] C. Y. Laporte and A. April, *Software Quality Assurance*. Wiley, 2018.

[119] W. Suryn, *Software Quality Engineering: A Practitioner’s Approach*. Wiley, 2013.

[120] D. Galin, *Software Quality: Concepts and Practice*. Wiley, 2018.

[121] “ISO / IEC 25010 : 2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models,” 2013.

[122] J. A. M. López, “Adaptación del proceso de desarrollo software para cumplimiento de la adecuación funcional según ISO/IEC 25000,” UNIVERSIDAD DE CASTILLA LA MANCHA ESCUELA SUPERIOR DE INFORMÁTICA, 2017.

[123] K. Bennett and V. Rajlich, “Software Maintenance and Evolution: a Roadmap,” 2000, pp. 73–87, doi: 10.1145/336512.336534.

[124] A. Rana, “Assessment of Maintainability Metrics of Object-Oriented Software System,” *ACM SIGSOFT Softw. Eng.*, vol. 36 (5), 2011, doi: 10.1145/2020976.2020983.

[125] P. Grubb and A. A. Takang, *Software Maintenance: Concepts and Practice*. World Scientific, 2003.

[126] S. Srivastava, “Software metrics and Maintainability Relationship with CK Matrix,” 2012.

[127] R. S. Pressman, *Ingeniería de Software, un enfoque práctico*. 2005.

[128] S. L. Pfleeger, *Software Engineering: Theory and Practice*. Prentice Hall, 1998.

[129] S. R. Chidamber and C. F. Kemerer, “A metrics suite for object oriented design,” *IEEE Trans. Softw. Eng.*, vol. 20, no. 6, pp. 476–493, 1994, doi: 10.1109/32.295895.

[130] J. R. Morales, “Mantenibilidad y evolutividad en el software libre y de código abierto,” Universidad Nacional de La Plata.

[131] J. Farfán and V. Vega, “Mantenibilidad del Software. Consideraciones para su especificación y validación,” 2019, doi: http://dx.doi.org/10.4067/S0718-33052020000400654.

[132] J. D. Erazo, A. S. Florez, and F. J. Pino, “Análisis y clasificación de atributos de mantenibilidad del software: una revisión comparativa desde el estado del arte,” *Entre*

Cienc. e Ing., vol. 10, pp. 40–49, 2016, [Online]. Available: http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S1909-83672016000100006&nrm=iso.

[133] M. Ribeiro, R. Q. Reis, and A. J. G. Abelém, "How to automatically collect oriented object metrics: A study based on systematic review," in *2015 Latin American Computing Conference (CLEI)*, 2015, pp. 1–12, doi: 10.1109/CLEI.2015.7359985.

[134] P. Becker *et al.*, *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 1999.

[135] K. Beck, *Smalltalk Best Practice Patterns*. Pearson Education, 1996.

[136] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*. Addison-Wesley, 2005.

[137] *Object-oriented and Classical Software Engineering*. McGraw-Hill Higher Education, 2010.

[138] D. N. Padhy, "A Systematic Literature Review of an Object Oriented Metrics Components: Case Study for Evaluation of Reusability Criteria," vol. IJARSE, 2017.

[139] J. Cowell, "Object-Oriented Programming," in *Essential Visual Basic 6.0 fast*, London: Springer London, 2000, pp. 133–139.

[140] A. Varela, H. Perez-Gonzalez, F. Martinez, and C. Soubervielle-Montalvo, "Source Code Metrics: A Systematic Mapping Study," *J. Syst. Softw.*, vol. 128, 2017, doi: 10.1016/j.jss.2017.03.044.

[141] S. K. Dubey and A. Rana, "Assessment of Usability Metrics for Object-Oriented Software System," *SIGSOFT Softw. Eng. Notes*, vol. 35, no. 6, pp. 1–4, Nov. 2010, doi: 10.1145/1874391.1874400.

[142] D. Sonal and G. Kaur, "Comparative Study of the Software Metrics for the complexity and Maintainability of Software Development," *Int. J. Adv. Comput. Sci. Appl.*, vol. 4, 2013, doi: 10.14569/IJACSA.2013.040925.

[143] R. Boken and P. K. Bhatia, "An Approach Toward Measurement of Reusability of Component-Based Software (CBS)," *Lect. Notes Networks Syst.*, vol. 302, pp. 133 – 145, 2022, doi: 10.1007/978-981-16-4807-6_14.

[144] A. Kurmangali, M. E. Rana, and W. Rahman, "Impact of Abstract Factory and Decorator Design Patterns on Software Maintainability: Empirical Evaluation using CK Metrics," 2022, pp. 517–522, doi: 10.1109/DASA54658.2022.9765083.

[145] N. Kaur and H. Singh, "An empirical assessment of threshold techniques to discriminate the fault status of software," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 8, pp. 6339 – 6353, 2022, doi: 10.1016/j.jksuci.2021.03.003.

[146] J. Singh, K. S. Dhindsa, and J. Singh, "Software quality improvement and validation using reengineering," *J. Eng. Res.*, vol. 9, no. 4 A, pp. 59 – 73, 2021, doi: 10.36909/jer.9481.

[147] M. Atifi, A. Mamouni, and A. Marzak, "A Comparative Study of Software Testing

Techniques," 2017, pp. 373–390, doi: 10.1007/978-3-319-59647-1_27.

[148] "Adecuación Funcional," *ISO/IEC 25010*. https://iso25000.com/index.php/normas-iso-25000/iso-25010/20-adecuacion-funcional.

[149] N. Anggraini, M. J. D. Putra, and N. Hakiem, "Development of an Islamic Higher Education Institution Tracer Study Information System and It's Performance Analysis using ISO/IEC 25010," in *2019 7th International Conference on Cyber and IT Service Management (CITSM)*, 2019, vol. 7, pp. 1–6, doi: 10.1109/CITSM47753.2019.8965356.

[150] A. Acharya and D. Sinha, "Assessing the Quality of M-Learning Systems using ISO/IEC 25010," *Accent Social and Welfare Society*, vol. 6, no. 3, pp. 67–75, 2013.

[151] N. A. Hasanah, L. Atikah, and S. Rochimah, "Functional Suitability Measurement Based on ISO/IEC 25010 for e-Commerce Website," in *2020 7th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, 2020, pp. 70–75, doi: 10.1109/ICITACEE50144.2020.9239194.

[152] G. Fournier, *Essential Software Testing: A Use-Case Approach*. Auerbach Publishers, Incorporated, 2017.

[153] J. Gulati, P. Bhardwaj, and B. Suri, "Comparative Study of Personality Models in Software Engineering," 2015, pp. 209–216, doi: 10.1145/2791405.2791445.

[154] G. Guimarães *et al.*, "A comparative study of psychometric instruments in software engineering," in *Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE*, 2021, vol. 2021-July, pp. 229 – 234, doi: 10.18293/SEKE2021-108.

[155] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic Mapping Studies in Software Engineering," *Proc. 12th Int. Conf. Eval. Assess. Softw. Eng.*, vol. 17, 2008.

[156] V. Benet and O. John, "Los Cinco Grandes Across Cultures and Ethnic Groups: Multitrait Multimethod Analyses of the Big Five in Spanish and English," *J. Pers. Soc. Psychol.*, vol. 75, pp. 729–750, 1998, doi: 10.1037//0022-3514.75.3.729.

[157] C. Odo, J. Masthoff, and N. Beacham, "Group Formation for Collaborative Learning: A Systematic Literature Review," 2019, pp. 206–212.

[158] N. Maqtary, A. Mohsen, and K. Bechkoum, "Group Formation Techniques in Computer-Supported Collaborative Learning: A Systematic Literature Review," *Technol. Knowl. Learn.*, vol. 24, no. 2, pp. 169–190, 2019, doi: 10.1007/s10758-017-9332-1.

[159] M. Fernández-Hawrylak, A. Sánchez Ibáñez, and D. Heras Sevilla, "Las actividades de enseñanza-aprendizaje en el Espacio Europeo de Educación Superior: Las actividades prácticas con herramientas web 2.0," *Acad. y Virtualidad*, vol. 13, no. 1 SE-Artículos, pp. 61–79, May 2020, doi: 10.18359/ravi.4260.

[160] S. Bell, "Project-Based Learning for the 21st Century: Skills for the Future," *Clear. House A J. Educ. Strateg. Issues Ideas*, vol. 83, no. 2, pp. 39–43, 2010, doi: 10.1080/00098650903505415.

[161] Microsoft Corporation, "Microsoft Teams." Microsoft Corporation, 2023, [Online].

Available: https://www.microsoft.com/es-co/microsoft-teams/group-chat-software.

[162] F. e Abreu and W. Melo, “Evaluating the impact of object-oriented design on software quality,” in *Proceedings of the 3rd International Software Metrics Symposium*, 1996, pp. 90–99, doi: 10.1109/METRIC.1996.492446.

[163] J. S. Madureira, A. S. Barroso, R. P. C. do Nascimento, and M. S. Soares, “An Experiment to Evaluate Software Development Teams by Using Object-Oriented Metrics,” in *Computational Science and Its Applications -- ICCSA 2017*, 2017, pp. 128–144.

[164] N. Davila and I. Nunes, “A systematic literature review and taxonomy of modern code review,” *J. Syst. Softw.*, vol. 177, p. 110951, 2021, doi: 10.1016/j.jss.2021.110951.

[165] R. Malhotra and A. Chug, “Software Maintainability: Systematic Literature Review and Current Trends,” *Int. J. Softw. Eng. Knowl. Eng.*, vol. 26, pp. 1221–1253, 2016, doi: 10.1142/S0218194016500431.

[166] U. L. Kulkarni, Y. R. Kalshetty, and V. G. Arde, “Validation of CK Metrics for Object Oriented Design Measurement,” in *2010 3rd International Conference on Emerging Trends in Engineering and Technology*, 2010, pp. 646–651, doi: 10.1109/ICETET.2010.159.

[167] T. R. Nair, B. Aravindh, and R. S. Rangasamy, “Design property metrics to maintainability estimation: a virtual method using functional relationship mapping,” *ACM SIGSOFT Softw. Eng. Notes*, vol. 35, pp. 1–6, 2010, doi: 10.1145/1874391.1874404.

[168] JetBrains, “IntelliJ IDEA,” *IntelliJ IDEA*. 2020, [Online]. Available: https://www.jetbrains.com/idea/.

[169] I. Sommerville and P. Sawyer, *Requirements Engineering: A Good Practice Guide*. Wiley, 1997.

[170] S. V Paunonen and M. Ashton, “Big five factors and facets and the prediction of behavior,” *J. Pers. Soc. Psychol.*, vol. 81, pp. 524–539, 2001, doi: 10.1037/0022-3514.81.3.524.

[171] L. R. Goldberg, “An alternative ‘description of personality’: The Big-Five factor structure.,” *Journal of Personality and Social Psychology*, vol. 59. American Psychological Association, US, pp. 1216–1229, 1990, doi: 10.1037/0022-3514.59.6.1216.

[172] B. Beizer, *Software Testing Techniques*. Van Nostrand Reinhold, 1990.

[173] B. Leijdekkers, “MetricsReloaded,” 2021. https://plugins.jetbrains.com/plugin/93-metricsreloaded.

[174] IBM Corporation, “IBM SPSS.” 2019, [Online]. Available: https://www.ibm.com/co-es/products/spss-statistics.

[175] S. S. Shapiro and M. B. Wilk, “An Analysis of Variance Test for Normality (Complete Samples),” *Biometrika*, vol. 52, no. 3/4, pp. 591–611, 1965, Accessed: Feb. 02, 2023. [Online]. Available: http://www.jstor.org/stable/2333709.

[176] L. Capretz, D. Varona, and A. Raza, “Influence of Personality Types in Software Tasks Choices,” *Comput. Human Behav.*, vol. 52, 2015, doi: 10.1016/j.chb.2015.05.050.

ANEXOS

Anexo 1. Formato de consentimiento informado empleado en el estudio.

UNIVERSIDAD DE NARINO. COMITE DE ETICA EN INVESTIGACIONES.
EFECTOS DE LA METICULOSIDAD COMO DIMENSIÓN DE LA PERSONALIDAD EN LOS ATRIBUTOS DE MANTENIBILIDAD Y ADECUACION FUNCIONALIDAD DEL SOFTWARE EN EQUIPOS DE DESARROLLO
CONSENTIMIENTO INFORMADO

Yo, _____ mayor de edad, identificado(a) con cc No _____ de _____, actuando en nombre propio, - (3) libre, espontáneamente y sin presiones indebidas.

DECLARO

Que he recibido toda la información clara y concreta en forma oral y escrita, por parte de Juan David Delgado Jojoa, el día ____ del mes de _____ del año _____, sobre el trabajo de investigación: **"Efectos de la meticulosidad como dimensión de la personalidad en los atributos de mantenibilidad y adecuación funcionalidad del software en equipos de desarrollo"** que realizarán a su cargo, en representación de Universidad de Nariño y el objetivo del proyecto es: evaluar la incidencia de la dimensión de meticulosidad en la calidad de producto software en equipos de desarrollo.

Me han advertido que en el proceso de investigación, en ningún momento se hará público mi nombre y/o documento de identificación, salvaguardando la confidencialidad de la información suministrada y mi privacidad, como tampoco saldrán a la luz pública hechos relacionados que puedan identificarme y sobre los cuales se guardarán siempre y en todo el estudio, todas las reservas y discrecionalidades correspondientes.

Me han explicado y he comprendido satisfactoriamente la naturaleza y propósito del estudio aludido en el que se incluirá un total de _____ investigados y de las posibles implicaciones que podría tener, especialmente que no corro ningún riesgo. He podido preguntar mis inquietudes al respecto y he recibido las respuestas y explicaciones en forma satisfactoria. También se me ha informado de mi derecho a participar voluntariamente en la investigación y la posibilidad de retirarme sin ningún tipo de consecuencias.

Se me ha informado que en caso de dudas, explicaciones adicionales o inconformidades de mi parte frente al estudio puedo comunicarme con el investigador principal. Juan David Delgado Jojoa al teléfono 3015335022.

He sido interrogado(a) sobre la aceptación o no, de esta autorización para este estudio, por lo tanto

AUTORIZO:

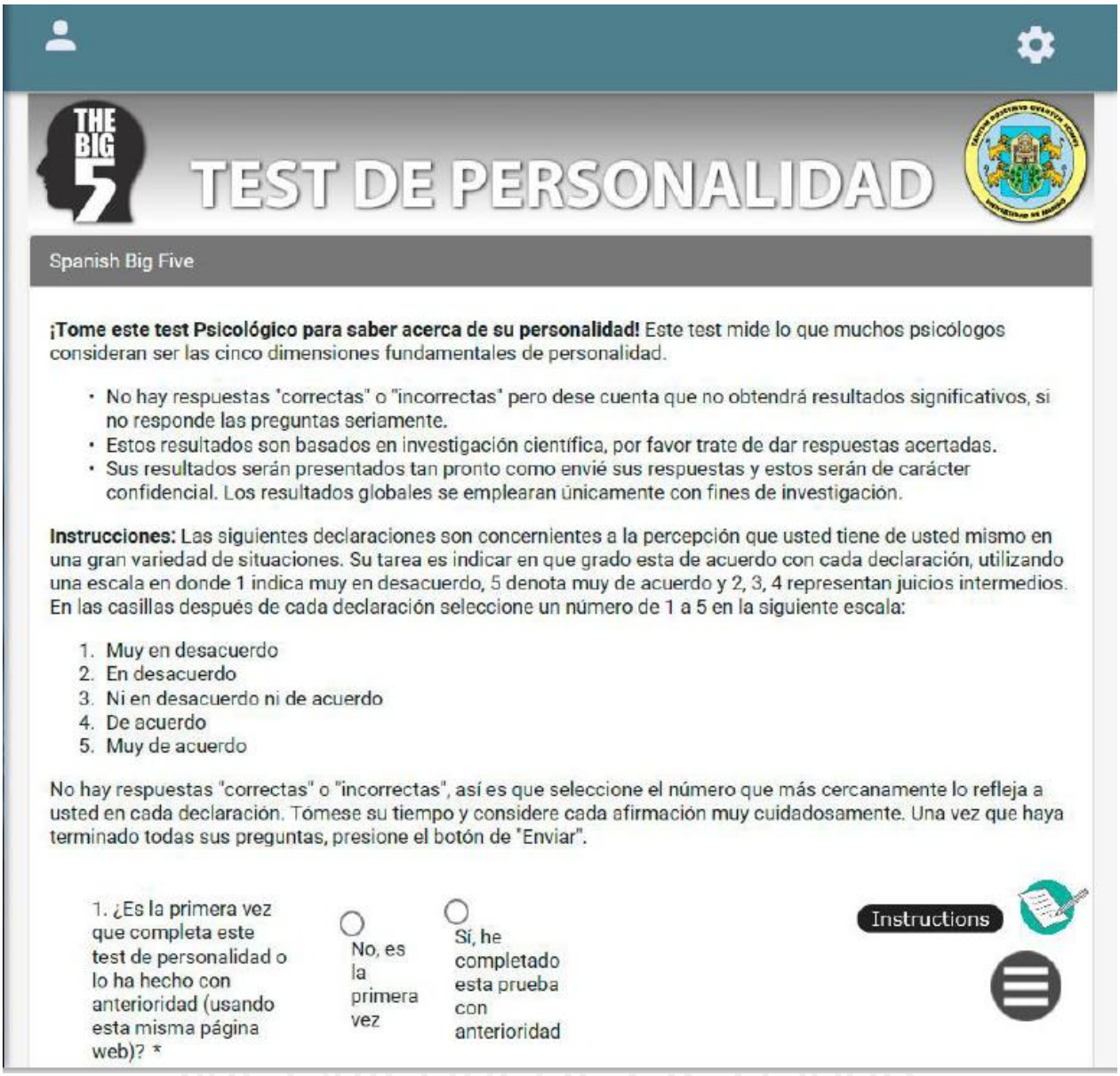
Para que Juan David Delgado Jojoa me realice una o varios cuestionarios que se solicita en la información que previamente se ha explicado su contenido y propósito. Esta autorización se concede por el término de (6 meses) a partir del día de la firma del presente consentimiento.

El grupo investigador se compromete a informarme de los resultados globales o parciales de la investigación, y/o de los que de manera positiva o negativa puedan influenciar en mi estado social o de salud

En constancia, se firma el presente documento, en dos copias, una para el investigador y otra para el investigado, con sus anexos (si los hay) en Pasto, a los _____días del mes de _____ del año _____

_____	_____
Nombre del participante	Firma y cédula del participante
_____	_____
Nombre del investigador	Firma y cédula del investigador

Anexo 2. Captura de pantalla de BF-Manager.



URL temporal de acceso: 192.168.185.26:81/moodle/

Anexo 3. Problema planteado para el desarrollo del producto software.

Como propietario de un taller mecánico de vehículos, se necesita un sistema que permita gestionar de forma automática todo lo relacionado con el mantenimiento y/o reparación de los vehículos de los clientes. Actualmente, la empresa maneja cuatro tipos de vehículos, que son: motos, automóviles, camionetas y camiones, pero necesito la flexibilidad de agregar nuevos tipos de vehículos en el futuro.

Además, el taller tiene varios tipos trabajadores, cada uno de los cuales tiene una especialidad única para cada tipo de vehículo. Debido a esto, los empleados ganan un salario diferente según su especialidad. Por otra parte, los clientes tienen dos opciones de pago, que son: pago en efectivo, tarjeta y pago a crédito. Sin embargo, si el vehículo solo requiere mantenimiento, el pago solo se puede hacer en efectivo.

En el caso de los clientes que requieren crédito, solo se les otorga si el total a pagar es igual o mayor a un millón de pesos. Además, en el taller también vende productos como aceite, cadenas, bombillas, etc., por lo que se necesita un pequeño módulo para gestionar el inventario de los repuestos y accesorios.

Para recompensar la lealtad de los clientes, aquellos que han llevado su vehículo al taller tres o más veces en el año en curso tendrán derecho a un descuento del 3% para el mantenimiento y del 5% para las reparaciones. Este beneficio no se podrá aplicar más de una vez en el año.

Finalmente, para fomentar la fidelidad de nuestros clientes, contamos con un módulo que permite sortear un mantenimiento gratuito para un cliente del taller.

A continuación, se presenta las historias de usuario para la construcción del producto:

Historia de usuario	
Número: 1	**Usuario:** Administrador
Nombre: Registro de usuarios	
Prioridad en negocio:	**Riesgo en el desarrollo:**
Puntos estimados:	**Iteración asignada:**
Descripción:	
Como administrador, quiero que el software me permita registrar usuarios para poder ingresar al sistema. El usuario debe tener los siguientes atributos: • Nombres (campo obligatorio) • Apellidos (campo obligatorio) • Cédula (campo obligatorio) • Dirección (campo obligatorio) • Teléfono (campo obligatorio) • Email (campo obligatorio) • Ciudad (campo obligatorio) • Fecha de nacimiento (campo obligatorio) • Nombre usuario (campo obligatorio) • Contraseña (campo obligatorio)	
Criterios de aceptación:	
• Antes de registrar al usuario, el sistema debe validar que el nombre de usuario sea único y que la contraseña tenga más de 6 caracteres. • El sistema debe manejar los roles de administrador y empleado.	

• Los usuarios con rol de administrador y empleado pueden acceder a todos los módulos y operar sobre ellos, excepto el de gestionar usuarios y empleados, que sólo podrá ser operado por el administrador. • El rol administrador puede actualizar, eliminar y ver todos los registros de los usuarios. • El rol empleado puede ver únicamente su información personal y sólo tiene permiso para actualizar su contraseña. • Para cambiar la contraseña del rol de empleado, el sistema debe pedirle que ingrese su contraseña anterior y, si es válida, permitirle actualizar su contraseña. De lo contrario, la actualización no será posible. • La contraseña debe estar cifrada utilizando un método de cifrado opcional. • El sistema debe registrar el usuario y verificar que la acción se haya ejecutado con éxito. • El sistema debe validar que los campos obligatorios se completen antes de registrar al usuario.
Observación:
Se debe asegurar la privacidad y seguridad de la información de los usuarios.

Fuente: esta investigación.

Historia de usuario	
Número: 2	**Usuario:** Administrador/Empleado
Nombre: Ingreso al sistema	
Prioridad en negocio:	**Riesgo en el desarrollo:**
Puntos estimados:	**Iteración asignada:**
Descripción:	
Como usuario, quiero que al ingresar al sistema se me solicite un nombre de usuario y una contraseña para garantizar la seguridad y privacidad de la información del taller.	
Criterios de aceptación:	
• El sistema debe validar que el nombre de usuario y la contraseña sean correctos antes de permitir el acceso al sistema. • El sistema debe mostrar un mensaje de error si el nombre de usuario y la contraseña son incorrectos y restar un intento. • El sistema debe permitir hasta 3 intentos fallidos de inicio de sesión antes de cerrar la ventana de login. • El sistema debe permitir el acceso al sistema si el nombre de usuario y la contraseña son correctos.	
Observación:	
La seguridad y privacidad de la información son aspectos clave para el correcto funcionamiento del sistema, por lo que es importante garantizar un ingreso seguro al mismo.	

Fuente: esta investigación.

Historia de usuario	
Número: 3	**Usuario:** Administrador/Empleado
Nombre: Gestionar Vehículos	
Prioridad en negocio:	**Riesgo en el desarrollo:**
Puntos estimados:	**Iteración asignada:**
Descripción:	
Como usuario del sistema, necesito poder registrar los vehículos que ingresan al taller. Para ello, el vehículo debe contar con los siguientes atributos obligatorios: • Placa (debe ser único) (campo obligatorio) • Tipo de vehículo (moto, automóvil, camioneta o camión) • Modelo (campo obligatorio) • Color (campo obligatorio) • Marca (campo obligatorio) • Observación	

Criterios de aceptación:
✓ Registrar vehículo y verificar que se ejecutó la acción con éxito ✓ Editar vehículo y verificar que se ejecutó la acción con éxito ✓ Validar campos obligatorios
Observación:

Fuente: esta investigación.

Historia de usuario	
Número: 4	**Usuario:** Administrador/Empleado
Nombre: Gestionar clientes	
Prioridad en negocio:	**Riesgo en el desarrollo:**
Puntos estimados:	**Iteración asignada:**
Descripción:	
Como usuario del sistema, necesito poder gestionar a los clientes para llevar un registro actualizado y completo de su información, lo que permitirá realizar la facturación y llevar un seguimiento de sus vehículos. El cliente debe contar con los siguientes atributos obligatorios: • Nombres (campo obligatorio) • Apellidos (campo obligatorio) • Cédula (campo obligatorio) • Dirección (campo obligatorio) • Teléfono (campo obligatorio) • Email • Ciudad (campo obligatorio) • Fecha de nacimiento (campo obligatorio) • Vehículo (campo obligatorio)	
Criterios de aceptación:	
• Antes de registrar al cliente, el sistema debe validar que al menos un vehículo ha sido asignado al cliente (HU-3). • Registrar cliente y verificar que se ejecutó la acción con éxito. • Editar cliente y verificar que se ejecutó la acción con éxito. • El sistema NO permitirá la eliminación de un cliente del sistema. • Validar campos obligatorios.	
Observación:	
Un cliente puede tener uno o más vehículos	

Historia de usuario	
Número: 5	**Usuario:** Administrador
Nombre: Gestionar empleados	
Prioridad en negocio:	**Riesgo en el desarrollo:**
Puntos estimados:	**Iteración asignada:**
Descripción:	
Como administrador, quiero gestionar a los empleados para llevar un historial y asignarlos a trabajos según su especialidad, lo que permitirá mejorar la eficiencia y calidad del servicio ofrecido por el taller. El empleado debe tener los siguientes atributos: • Nombres (campo obligatorio) • Apellidos (campo obligatorio) • Cédula (campo obligatorio) • Dirección (campo obligatorio) • Teléfono (campo obligatorio) • Email • Ciudad (campo obligatorio) • Fecha de nacimiento (campo obligatorio) • Fecha Ingreso (campo obligatorio) • Especialidad (moto, automóvil, camioneta o Camión) • Sueldo(campo obligatorio)	
Criterios de aceptación:	
• Registrar al empleado y verificar que se ejecutó la acción con éxito • Editar al empleado y verificar que se ejecutó la acción con éxito • El empleado NO podrá ser eliminado • Asignar empleados a trabajos según su especialidad y verificar que se ejecutó la acción con éxito • Validar campos obligatorios	
Observación:	

Fuente: esta investigación.

Historia de usuario	
Número: 6	**Usuario:** Administrador/Empleado
Nombre: Registro planilla de ingreso de vehículos	
Prioridad en negocio:	**Riesgo en el desarrollo:**
Puntos estimados:	**Iteración asignada:**
Descripción:	
Como usuario, quiero registrar el ingreso de los vehículos que necesitan mantenimiento o reparación para poder facturar el trabajo realizado y llevar un historial de las actividades. La planilla de ingreso de vehículos debe contener los siguientes campos: • Fecha ingreso (campo obligatorio) • Fecha entrega • Nombre cliente (campo obligatorio) • Placa vehículo (campo obligatorio) • Tipo de trabajo (mantenimiento o reparación) • Nombre empleado quien va a realizar el trabajo (campo obligatorio) • Descripción del trabajo a realizar (campo obligatorio) • Observación	
Criterios de aceptación:	

• La fecha de ingreso no será editable y se asignará automáticamente con la fecha del sistema al registrar la planilla. • La fecha de entrega se actualizará automáticamente con la fecha del sistema solo después de haber facturado la planilla de forma exitosa • Al ingresar la placa del vehículo, el sistema debe verificar que el cliente tiene al menos un vehículo registrado (HU-4) y que el vehículo no tiene una planilla de ingreso activa • Al seleccionar el tipo de vehículo (moto, automóvil, camioneta o camión), se debe mostrar una lista desplegable con los empleados que tienen la especialidad correspondiente • No se validará si el empleado ya fue asignado a otra planilla de trabajo • Se validará que los campos obligatorios estén completos • Registrar planilla y verificar que se ejecutó la acción con éxito • Editar planilla y verificar que se ejecutó la acción con éxito
Observación:

Fuente: esta investigación.

Historia de usuario	
Número: 7	**Usuario:** Administrador/Empleado
Nombre: Gestionar productos	
Prioridad en negocio:	**Riesgo en el desarrollo:**
Puntos estimados:	**Iteración asignada:**
Descripción:	
Como usuario, quiero poder gestionar los productos que ofrece el taller para llevar un inventario de forma sencilla. Los productos deben tener la siguiente información: • Fecha (campo obligatorio) • Código (campo obligatorio, debe ser único) • Nombre Producto (campo obligatorio) • Categoría del producto (Solo maneja dos categorías: repuesto y accesorio) • Precio compra (campo obligatorio) • precio venta (campo obligatorio) • Stock (campo obligatorio)	
Criterios de aceptación:	
• Registrar un producto y verificar que se ha ejecutado la acción con éxito • Editar un producto y verificar que se ha ejecutado la acción con éxito • Eliminar un producto y verificar que se ha ejecutado la acción con éxito • Validar que los campos obligatorios se han completado correctamente	
Observación:	

Fuente: esta investigación.

Historia de usuario	
Número: 8	**Usuario:** Administrador/Empleado
Nombre: Facturación	
Prioridad en negocio:	**Riesgo en el desarrollo:**
Puntos estimados:	**Iteración asignada:**
Descripción:	
Como usuario, quiero facturar la planilla ingresada (HU-6) para llevar la contabilidad del taller de forma organizada y rápida. La factura debe contar con la siguiente información:	

• Fecha (campo obligatorio) • Número de factura (campo obligatorio) • Nombre del cajero (campo obligatorio) • Placa del Vehículo (campo obligatorio) • Tipo vehículo (campo obligatorio) • Nombre del cliente (campo obligatorio) • Tipo de trabajo (mantenimiento o reparación) (campo obligatorio) • Nombre del empleado quien realizó el trabajo (campo obligatorio) • Descripción del trabajo realizado (campo obligatorio) • Ítems productos (Solo en caso de que el cliente lleve uno o más productos) • Total a pagar (campo obligatorio) • Descuento • Forma de pago (efectivo, tarjeta o crédito:) (campo obligatorio) • Observación
Criterios de aceptación:
✓ La fecha no debe ser editable y se debe tomar del sistema ✓ El número de factura puede ser automático o manual, pero debe ser único ✓ Para efectuar la factura, primero se debe realizar una búsqueda en la planilla (HU-6) con la placa del vehículo para obtener información del tipo de trabajo, vehículo, cliente, etc. ✓ En el resultado de la búsqueda anterior, se debe tener en cuenta el valor de la "fecha de entrega"; si esta está en nulo, se continúa con el proceso de facturación, de lo contrario se debe mostrar el siguiente mensaje: "Este trabajo ya fue facturado", y no se permitirá facturar. ✓ En caso de no encontrar ninguna información con la placa del vehículo, se debe mostrar el siguiente mensaje. "No se encontró ningún registro con esa placa" y no se permitirá facturar. ✓ El usuario tiene la opción de buscar y agregar productos para vender al cliente si este lo solicita, y esto se sumará al total a pagar. ✓ Si el stock del producto está en cero (0) no podrá agregarlo, y mostrará el siguiente mensaje: "el producto está agotado" ✓ Si realiza la venta de productos, estos se deben descontar al stock actual ✓ Las opciones de pago son efectivo, tarjeta y crédito ✓ Si el tipo de trabajo es de mantenimiento no tendrá la opción de pagar a crédito ✓ Si el cliente ha llevado su vehículo al taller igual o superior a 3 veces en el año en curso, se realizará un descuento de la siguiente forma: por mantenimiento un descuento del 3% y por reparo un 5% del total a pagar. Cuando se aplique este descuento se debe tener en cuenta para no volver aplicar este beneficio en el mismo año. ✓ El cliente puede obtener un crédito sólo si el total a pagar es igual o mayor a un millón de pesos y el trabajo fue una reparación ✓ Si el cliente decide pagar a crédito se debe llevar a cabo lo descrito en la siguiente HU-9 ✓ Una vez registrada la factura con éxito, se debe actualizar con la fecha actual del sistema el valor del campo "fecha entrega" de la planilla facturada. ✓ La factura NO tendrá la opción de editar y tampoco de eliminar una vez se haya registrado
Observación:

Fuente: esta investigación.

Historia de usuario	
Número: 9	**Usuario:** Administrador/Empleado
Nombre: Facturación a crédito	
Prioridad en negocio:	**Riesgo en el desarrollo:**
Puntos estimados:	**Iteración asignada:**
Descripción:	
Como usuario, quiero almacenar las ventas que fueron realizadas a crédito (HU 8)	

El tipo de pago a crédito debe contar la siguiente información: • Número de factura (campo obligatorio) • Total a financiar (campo obligatorio) • Tiempo a financiar (campo obligatorio) • tasa de interés (campo obligatorio) • interés (campo obligatorio)
Criterios de aceptación:
• El número de factura corresponderá al que se generó en la HU-8 • El Total a financiar corresponderá al total a pagar que se generó en la HU-8 • El Tiempo a financiar debe estar entre 1 a 5 años • La tasa de interés tiene un valor fijo del 5% • El valor del campo de interés será el resultado de aplicar la formula interés simple: I = C (Total a financiar) * R (tasa interés) * T (Tiempo a financiar)
Observación:
No se realizarán cobros mensuales y ningún otro tipo de operaciones, solamente se necesita almacenar la información de la facturación a crédito.

Fuente: esta investigación.

Historia de usuario	
Número: 10	**Usuario:** Administrador/Empleado
Nombre: Sorteo mantenimiento vehículo	
Prioridad en negocio:	**Riesgo en el desarrollo:**
Puntos estimados:	**Iteración asignada:**
Descripción:	
Como usuario, quiero realizar un sorteo de un mantenimiento gratuito para cualquier tipo de vehículo con el objetivo de mantener la clientela y llevar un historial de los sorteos. El sorteo debe tener la siguiente información: • Fecha (campo obligatorio) • Nombre cliente (campo obligatorio) • Tipo de vehículo (campo obligatorio) • Premio (campo obligatorio)	
Criterios de aceptación:	
✓ Para este sorteo, se necesita solamente de un botón, cuando al darle clic a este botón, se debe seleccionar de forma aleatoria un cliente de la tabla clientes. ✓ El sorteo se realizará en dos instancias, la primera se realizará el 30 de enero y la última el 15 de diciembre de cada año ✓ Si no está en esas fechas mencionadas, el botón debe estar deshabilitado ✓ El cliente ganador ya no podrá volver a participar nunca más	
Observación:	

Anexo 4. Diseño casos de prueba.

Número de caso de prueba	Descrip-ción	Entrada	Resultado esperado	Resultado obtenido	Se imple-mentó (COMF)	El resultado correcto (CORF)	satisface las necesidades y objetivos del usuario (PERF)
1	Inicio de sesión al sistema	Usuario y contra-seña	Inicio de sesión exitoso y redirec-ción a la página principal	Inicio de sesión exitoso y redirec-ción a la página principal	S	S	S
2	Editar con-traseña	Datos de usuario	Actuali-zar la contra-seña con éxito	Falló al actualizar la contra-seña	S	N	N
…….. …….. ……..	…….. …….. ……..	…….. …….. ……..	…….. …….. ……..	…….. …….. ……..	…….. …….. ……..	…….. …….. ……..	…….. …….. ……..

Fuente: esta investigación.

Casos de prueba se crean para verificar que la funcionalidad del software cumple con los criterios de aceptación definidos en las Historias de Usuario.

Para puntuar la COMF se realizó de la siguiente forma:

$$X = 1 - \frac{A}{B}$$

A = Total de número de funciones faltantes

B = Total de número de funciones especificadas

$$COMF = 1 - \frac{0}{2} => 1$$

Para este ejemplo, se supone que se especificaron un total de 2 funciones, de las cuales todas se implementaron según se muestra en la tabla anterior. Por lo tanto, el resultado es de 1, lo que indica que se cumplió satisfactoriamente con todas las funciones especificadas. Es importante tener en cuenta que, entre más cercano a 1 sea el puntaje, mejor será la evaluación del software en cuanto a la implementación de las funciones requeridas.

Para puntuar la corrección funcional, teniendo en cuenta la formula anterior, se realizó de la siguiente forma:

A = Total de número de funciones incorrectas

B = Total de número de funciones implementadas

$$CORF = 1 - \frac{1}{2} => 0{,}5$$

Para puntuar la PERF, teniendo en cuenta la formula anterior, se realizó de la siguiente forma:

A = Total de número de funciones adecuadas para lograr los objetivos del usuario

B = Total de número de funciones implementadas

$$PERF = 1 - \frac{1}{2} => 0{,}5$$

Para garantizar que el software cumpla con los requisitos y expectativas del cliente, se realiza el diseño de casos de prueba para cada Historia de Usuario en el proceso de pruebas. Y se aplica la fórmula que se mencionó anteriormente, que permite obtener puntajes para variables clave como la completitud, corrección y PERF. Al aplicar esta fórmula, se puede evaluar de manera cuantitativa el rendimiento del software en términos de la implementación de las funciones y criterios de aceptación definidos en cada HU. Esto permite identificar y solucionar rápidamente cualquier problema o error que pueda afectar la calidad del software y su capacidad para cumplir con los requisitos del cliente.

Anexo 5. Ponencia realizada en el VIII congreso Iberoamericano de HCI.

13, 14 y 15 de octubre de 2022

HCI
VIII Jornadas Iberoamericanas de Interacción Humano-Computador

El Comité Organizador certifica que la ponencia

"Modelos e instrumentos psicológicos empleados para identificar los rasgos de personalidad de los desarrolladores de software: Un estudio de mapeo sistemático"

Juan Delgado J, Oscar Revelo Sánchez and Sandra Vallejo

ha sido presentada como parte de las actividades

científicas del evento.

Dr. C. Cesar Alberto Collazo
Presidente Jornadas Iberoamericanas HCI

Dr. C. Omar Correa Madrigal
Coordinador Local

UCI Universidad de las Ciencias Informáticas

HCI COLLAB

Anexo 6. Artículo publicado en el libro HCI-COLLAB 2022 de la editorial Springer.

Springer Link

Search Log in

Iberoamerican Workshop on Human-Computer Interaction

↳ HCI-COLLAB 2022: **Human-Computer Interaction** pp 146–161 | Cite as

Home › Human-Computer Interaction › Conference paper

Psychological Models and Instruments Employed to Identify Personality Traits of Software Developers: A Systematic Mapping Study

Juan David Delgado Jojoa, Oscar Revelo Sánchez & Sandra Vallejo Chamorro

Conference paper | First Online: 22 January 2023

38 Accesses

Part of the Communications in Computer and Information Science book series (CCIS,volume 1707)

Abstract

Over the years, a significant amount of research has been done to discover links and/or effects of human personality in the field of software engineering (SI). It has been shown that personality has a positive influence on the different tasks and/or processes in this field. Related studies helped to identify a suitable personality type for each profile in the SI. However, contradictions have also been found in the results that question their validity, therefore, generalizing and applying these results may be misleading. This is because one of the key and complex factors is evaluating the personality of engineers or software developers from a psychological point of view. Through a systematic mapping study, an attempt was made to

Access via your institution

Chapter USD 29.95

Price excludes VAT (Colombia)

- DOI: 10.1007/978-3-031-24709-5_11
- Chapter length: 16 pages
- Instant PDF download
- Readable on all devices
- Own it forever
- Exclusive offer for individuals only
- Tax calculation will be finalised during checkout

Buy Chapter

eBook USD 64.99

Softcover Book USD 84.99

Learn about institutional subscriptions

Sections | Figures | References

Abstract

Se puede consultar en:

https://link.springer.com/chapter/10.1007/978-3-031-24709-5_11

Anexo 7. Artículo publicado en la Revista Ciencia e Ingeniería Neogranadina.

Se puede consultar en:

https://revistas.unimilitar.edu.co/index.php/rcin/article/view/6872