UNIVERSIDAD
NACIONAL
DE COLOMBIA

Diagnóstico de la estimación de esfuerzo en métodos ágiles para desarrollo de software en Colombia

Diana Marcela Gómez Cardozo

Universidad Nacional de Colombia

Facultad de Ingeniería, Departamento de Ingeniería de sistemas e Industrial

Bogotá D.C, Colombia

2022

Diagnóstico de la estimación de esfuerzo en métodos ágiles para desarrollo de software en Colombia

Diana Marcela Gómez Cardozo

Trabajo de investigación presentado como requisito para optar al título de:

Magister en Ingeniería de Sistemas y Computación

Director:

Jairo Hernán Aponte Melo, Ph.D

Línea de Investigación:

Métodos y Tecnologías para el Desarrollo de Software

Grupo de Investigación:

Colectivo de Investigación en Ingeniería de Software - ColSWE

Universidad Nacional de Colombia

Facultad de Ingeniería, Departamento de Ingeniería de sistemas e Industrial

Bogotá D.C, Colombia

2022

Declaración de obra original

Yo declaro lo siguiente:

He leído el Acuerdo 035 de 2003 del Consejo Académico de la Universidad Nacional. «Reglamento sobre propiedad intelectual» y la Normatividad Nacional relacionada al respeto de los derechos de autor. Esta disertación representa mi trabajo original, excepto donde he reconocido las ideas, las palabras, o materiales de otros autores.

Cuando se han presentado ideas o palabras de otros autores en esta disertación, he realizado su respectivo reconocimiento aplicando correctamente los esquemas de citas y referencias bibliográficas en el estilo requerido.

He obtenido el permiso del autor o editor para incluir cualquier material con derechos de autor (por ejemplo, tablas, figuras, instrumentos de encuesta o grandes porciones de texto).

Por último, he sometido esta disertación a la herramienta de integridad académica, definida por la universidad.

Diana Marcela Gómez Cardozo

_____

Nombre

Fecha: 28/08/2022

Resumen

Diagnóstico de la estimación de esfuerzo en métodos ágiles para desarrollo de software en Colombia

La estimación del esfuerzo de desarrollo de software (EEDS) es fundamental para planificar y controlar proyectos de software y crítica para su éxito. La práctica ha demostrado que el establecimiento de objetivos realistas, estimaciones precisas y una buena planificación y control son actividades esenciales para el éxito del proyecto. Además, el Desarrollo Ágil de Software (DAS) es hoy en día el modelo más utilizado para organizar todas las actividades requeridas para la construcción de software. Este trabajo de investigación se centra en conocer cómo los profesionales colombianos realizan estimaciones de esfuerzo dentro de proyectos de desarrollo ágil de software. Para este propósito, se realizó un estudio del estado del arte académico en temas relacionados con estimación de software para desarrollos ágiles dentro de Colombia y otros países, con el objetivo de poder comparar nuestras prácticas con las de otras regiones. Posteriormente se llevó a cabo una encuesta exploratoria realizada a profesionales colombianos con experiencia en estimación de esfuerzo para desarrollo ágil. El diseño de la encuesta se basó en los estudios similares identificados a nivel mundial con el fin de obtener resultados comparables. Esta encuesta fue el instrumento utilizado para recopilar información, mientras que una combinación de análisis cualitativo y cuantitativo se utilizó para interpretar los resultados; se pretende así conocer el estado de la práctica en Colombia con relación a las técnicas utilizadas para estimar esfuerzo en desarrollos ágiles, la medición de la exactitud de dichas estimaciones, perfil profesional y herramientas en las que se apoyan los participantes del estudio para la elaboración de sus estimaciones. Por último, se realiza una comparación de estos resultados, con los hallazgos de trabajos relacionados en otros lugares del mundo.

Palabras clave: Desarrollo ágil de software, Estimación de esfuerzo para desarrollo ágil de software, Métodos de estimación, Estimación de esfuerzo, Estudio empírico, Gestión de proyectos de software.

Abstract

Diagnosis of effort estimation in agile methods for Software development in Colombia

Software Development Effort Estimates (SDEE) are critical to planning and controlling software projects and fundamental to their success. Practice has shown that setting realistic goals, accurate estimates, and good planning and control are essential activities for project success. Furthermore, Agile Software Development (ASD) is today the most widely used model to organize all the activities required for software construction. This research work focuses on knowing how Colombian professionals make effort estimates within agile software development projects. For this purpose, a study of the academic state of the art was carried out on issues related to software estimation for agile developments within Colombia and other countries, with the objective of being able to compare our practices with those of other regions. Subsequently, an exploratory survey was carried out among Colombian professionals with experience in effort estimation for agile development. The survey design was based on similar studies identified globally in order to obtain comparable results. This survey was the instrument used to collect information, while a combination of qualitative and quantitative analysis was used to interpret the results; In this way, it is intended to know the state of the practice in Colombia in relation to the techniques used to estimate effort in agile developments, the measurement of the accuracy of such estimates, professional profile, and tools on which the study participants rely for estimating effort. Finally, a comparison of these results is made with the findings of related works in other parts of the world.

Keywords: Agile software development, Effort estimation for agile software development, Estimation methods, Effort estimation, Empirical study, Software project management.

Contenido

Lista de figuras

Pág.

Lista de Tablas

Pág.

Lista de abreviaturas

Abreviatura	Descripción
EEDS	Estimación del Esfuerzo de Desarrollo de Software
SDEE	Software Development Effort Estimates
DAS	Desarrollo Ágil de Software
ASD	Agile Software Development
PMBOK	Project Management Body of Knowledge
XP	Extreme Programming
TDD	Test Driven Development
BDD	Behavior Driven Development
LSD	Lean Software Development
ASD	Adaptative Software Development
FDD	Feature Driven Development
SAFe	Scaled Agile Framework
DSDM	Dynamic Systems Development Method
SDEE	Software Development Effort Estimation
AEE	Agile Effort Estimation
SP	Story Points
EDT	Estructura de Descomposición del Trabajo
WBS	Work Breakdown Structure
COCOMO	Constructive Cost Model
FP	Function Points
ML	Machine Learning
CBR	Case Based Reasoning
ANN	Artificial Neural Networks
MMRE	Magnitud Media de Error Relativo
PRED (r)	Percentage Relative Error Deviation
ANOVA	Analysis Of Variance
SLR	Systematic Literature Review
MinTIC	Ministerio de Tecnologías de la Información y las Comunicaciones

Introducción

La estimación del esfuerzo de desarrollo de software es el proceso de evaluar o predecir el esfuerzo de construcción más realista, duración y costo de producto de software [1]. En el PMBOK [2] se define la estimación de esfuerzo como una evaluación de la cantidad o resultado probable. Normalmente se aplica a los costos, recursos, esfuerzo y duración del proyecto.

La evidencia acumulada a lo largo del tiempo ha demostrado que la estimación del esfuerzo es uno de los principales aspectos que determinan el éxito o fracaso de los proyectos de software.

El desarrollo ágil de software es un enfoque basado en la construcción de software iterativo e incremental, entrega temprana de valor comercial, participación permanente del usuario y respuesta rápida a los cambios [3].

Los primeros métodos de desarrollo ágil surgieron a mediados de la década de 1990 y se centraron en responder a los cambios de forma rápida y eficiente mediante el uso de plazos cortos, iteraciones con refinamientos flexibles y planes y objetivos cambiantes [4]. Dentro del contexto general de ingeniería del software, la estimación del esfuerzo se ha estudiado ampliamente, pero debido a la naturaleza iterativa de los métodos ágiles, se requieren técnicas de evaluación especializadas [5]. Las estimaciones de esfuerzo en desarrollo ágil están lejos de ser perfectas y en la mayoría de los casos la tendencia es a subestimar el esfuerzo requerido. Los requisitos, la gestión y los problemas relacionados con el equipo se citan como las principales causas de las grandes diferencias entre el esfuerzo real y el estimado [6].

El objetivo principal de este trabajo es conocer y realizar un análisis del estado actual de la práctica de estimación de software para desarrollo ágil en un contexto local, es decir, en varias regiones de Colombia. De igual forma verificar los principales hallazgos de otros estudios relacionados con el proceso de estimación del esfuerzo en los proyectos de

desarrollo ágil de software, específicamente sobre los métodos utilizados, los problemas existentes y las oportunidades de mejora.

Para realizar el análisis del estado actual del proceso de estimación del software, se recopilaron y analizaron datos de cinco áreas principales de interés: (1) métodos utilizados para realizar la estimación del esfuerzo; (2) nivel de precisión de la estimación del esfuerzo en desarrollos ágiles; (3) predictores de esfuerzo utilizados en este proceso; (4) conjuntos de datos utilizados para realizar actividades de estimación de esfuerzo y (5) diferencias entre los hallazgos de este estudio y estudios previos realizados en otras regiones.

El instrumento de recolección de datos aplicado en este estudio fue una encuesta en línea. Esto permitió recopilar información relacionada con varios aspectos concernientes a la estimación del esfuerzo, junto con la información demográfica de las personas que juegan varios roles técnicos y gerenciales en desarrollo ágil de software.

El documento se presenta de la siguiente manera. La primera sección contempla las definiciones necesarias para enmarcar y comprender el contexto de este trabajo.

La segunda sección comprende la metodología de investigación aplicada para este proyecto. La tercera sección presenta los resultados recolectados y las amenazas a la validez. Por último, en la sección cuatro se presenta las conclusiones.

1. Capítulo 1: Marco conceptual

En esta sección se presentan conceptos básicos relacionados con las herramientas que se usaron para llevar a cabo este trabajo de investigación, las metodologías ágiles más comunes y los métodos de estimación de esfuerzo que pueden ser aplicados dentro de las mismas.

1.1 Investigación por encuesta

La investigación mediante encuestas se ha generalizado en el mundo occidental moderno y más allá. Una encuesta puede ser vista como una herramienta de investigación crucial en la academia, el gobierno y el sector privado [7].

La encuesta se trata de un sistema completo de recolección de información para describir, comparar o explicar conocimientos, actitudes y comportamiento. [8]
Las encuestas están diseñadas para producir estadísticas sobre una población objetivo. El proceso por el cual esto se hace se basa en inferir las características de la población objetivo de las respuestas proporcionadas por una muestra de encuestados. [9]

Como describen Wohlin et al. [10], el objetivo de una encuesta puede ser i) descriptivo, ii) explicativo, o iii) exploratorio. Descriptivo significa que sirve de apoyo al investigador para hacer afirmaciones sobre la población. Explicativo significa que ayudan al investigador a explicar tendencias, fenómenos o problemas observados en la población.
Exploratorio significa que ayudan al investigador a abrir nuevos caminos y descubrir nuevos conocimientos sobre un área que es hasta cierto punto desconocida.

Linåker et al. [11] plantean una serie de actividades fundamentales que se deben llevar a cabo en estudios basados en encuestas, dentro del campo de la Ingeniería del software. Como primer paso, se sugiere formular las preguntas de investigación, que se definen en función de la necesidad de investigación, e interés del investigador. Con base en las preguntas de investigación, se determina la población objetivo y la audiencia del estudio. La población objetivo es en la mayoría de los casos grande, lo que significa que es

imposible hacer preguntas a cada uno de los miembros, por tanto, se debe decidir una muestra de la población como siguiente paso.

Antes de abordar la población de muestra, se debe formular un cuestionario y luego evaluarlo en un subconjunto de la muestra de población para determinar si realmente es útil al propósito general de la investigación.

Cuando la población de la muestra ha respondido las preguntas del instrumento, el investigador puede analizar los datos y sacar conclusiones basadas en el análisis. Después de eso, el estudio con sus resultados y conclusiones puede ser informado.

1.1.1 Recomendaciones para el diseño de encuestas

Basado en una combinación de trabajos de diferentes autores, Klagge [12] resume una serie de recomendaciones para tener en cuenta antes, durante y después de la ejecución de una encuesta. Estas recomendaciones son:

Actividades previas a la encuesta:

Antes de realizar una encuesta, se deben completar los siguientes pasos:

- Desarrollar una declaración de objetivo clara y concisa: lo que el solicitante o investigador quiere saber y por qué quiere saberlo.
- Desarrollar las preguntas para el instrumento: deben ser claras, inequívocas y libres de prejuicios.
- Probar las preguntas en un grupo de voluntarios para determinar que la validez de los ítems es adecuada. Realizar los cambios necesarios en los elementos.
- Desarrollar la introducción a la encuesta y las cartas que precederán a la distribución de la encuesta, acompañar la distribución de la encuesta, y los recordatorios después de que se haya enviado la encuesta.
- Determinar la modalidad de distribución de la encuesta.
- Programar el envío de la carta inicial, el lanzamiento del instrumento de encuesta,
- y el envío de la(s) carta(s) de seguimiento.
- Evitar el “error de cobertura” recopilando una lista de muestra de posibles participantes que coincida lo más posible con la población de interés.

Distribución de la encuesta:

Los siguientes son los pasos generales que deben seguirse durante la distribución de un instrumento de encuesta.

- Evitar el "error de falta de respuesta" empleando técnicas de investigación recientemente utilizadas (estado del arte) en estudios por encuestas.
- Cada participante debe ser contactado por lo menos cuatro veces. El primer contacto será una carta (email) de presentación incluyendo la información especificada anteriormente y anunciando que la encuesta se realizará en breve. El segundo contacto será el envío de la propia encuesta. El tercer contacto será un recordatorio. El cuarto contacto será el recordatorio final.
- Cada contacto debe ser y verse diferente ya que la diferencia llama la atención. El momento de los contactos será determinado por la naturaleza de la población de encuestados y el medio utilizado.

Actividades posteriores a la encuesta:

Los pasos a continuación deben tomarse inmediatamente después de la fase de envío y recepción de resultados de la encuesta.

- Cada encuesta recibida será examinada para ver si está completa.
- Se examinará la totalidad de la base de datos para verificar que esté completa.
- Se tomarán decisiones sobre cómo manejar las encuestas incompletas y los campos de la base de datos.

1.2 Metodologías ágiles

El Desarrollo Ágil de Software (ASD, *Agile Software Development*) es un grupo de métodos de construcción de software basados en el desarrollo iterativo e incremental, donde los requisitos y las soluciones evolucionan a través de la colaboración entre equipos auto-organizados y multifuncionales. ASD promueve la planificación adaptativa, desarrollo evolutivo y entrega con un enfoque iterativo de tiempo limitado, además alienta las respuestas flexibles al cambio [13]. ASD se ha posicionado como el enfoque más importante dentro del desarrollo de software en los últimos años. [14]

El Manifiesto Ágil, formulado en 2001 por un grupo de expertos, estableció los valores y principios clave detrás de la filosofía ágil y ha tenido un gran impacto en la industria del software en todo el mundo [15].

Uno de los aspectos en común en las metodologías ASD existentes, es el énfasis en que el desarrollo de software ágil debe ser flexible y no estrictamente predictivo, dado que éste estará sujeto a una serie de cambios, en gran parte impredecibles al inicio del desarrollo del software. Por lo tanto, el desarrollo ágil requiere de ciclos de inspección y adaptación con una constante realimentación [16].

En esta sección se describe qué es el desarrollo ágil de software y cuáles son las metodologías ágiles más conocidas.

1.2.1 Manifiesto ágil

El manifiesto ágil [15] fue publicado en 2001, cuando un grupo de profesionales de la industria del software se reunió para establecer una alternativa a los procesos de desarrollo tradicionalmente aplicados. En dicha reunión se adoptó el término “Métodos Ágiles” para definir a los métodos que estaban surgiendo como alternativa a las metodologías que se consideraban densas y rígidas por su carácter normativo y dependencia de planes detallados previos al desarrollo.

En este manifiesto se determinaron cuatro postulados o valores que acogen a su vez doce principios.

Individuos e interacciones por encima de procesos y herramientas	Software funcionando por encima de documentación exhaustiva	Respuesta ante el cambio por encima de seguir un plan	**Colaboración con el cliente por encima de negociación contractual**
• Las personas son importantes por su capacidad para ser creativas e innovar. Los procesos y las herramientas deben servir de apoyo para el trabajo de las personas, pero no ser más importantes que ellas.	• El Manifiesto Agile no descarta o elimina la documentación del software , pero defiende que genera menos valor que el software en funcionamiento. Por lo que se debería reducir a lo mínimo indispensable.	• Las metodologías ágiles promueven la anticipación y la adaptación, frente a la planificación y el control que proponen las fórmulas de gestión tradicionales, en donde ajustar el mismo plan establecido inicialmente tal vez no se se.	• La colaboración y las buenas relaciones con el cliente genera más valor que el cumplimiento estricto de un contrato, que puede llegar a crear barreras y delimitar responsabilidades en los escenarios en los que se requiere más de lo obligatorio.

Descritos los cuatro valores, los autores de este manifiesto redactaron los siguientes, como los principios que se derivan de dichos valores:

- La mayor prioridad es satisfacer al cliente a través de la entrega temprana y continua de software con valor.
- Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
- Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al período de tiempo más corto posible.
- Los responsables del negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
- Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
- El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
- El software que funciona es la medida principal de progreso.
- Los procesos ágiles promueven el desarrollo sostenido. Los promotores, desarrolladores y usuarios debemos mantener un ritmo constante de forma indefinida.
- La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.
- La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
- Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
- A intervalos regulares, el equipo reflexiona sobre cómo ser más efectivo para, a continuación, ajustar y perfeccionar su comportamiento en consecuencia.

1.2.2 Metodologías ágiles más usadas

Como resultado de la revisión de la literatura realizada durante este estudio, se encontró que Scrum, Kanban y Extreme Programming son las metodologías ágiles más usadas [17]; sin embargo, existen otras también importantes pero que son aplicadas en menor proporción en el sector de desarrollo de software. Para efectos de este trabajo de investigación se consideraron las siguientes metodologías a la hora de indagar qué tanto y de qué forma son utilizadas por los profesionales colombianos en desarrollo de software. Al final de esta sección se presenta un resumen de las prácticas que se llevan a cabo en cada una de las metodologías descritas a continuación.

1.2.2.1 SCRUM

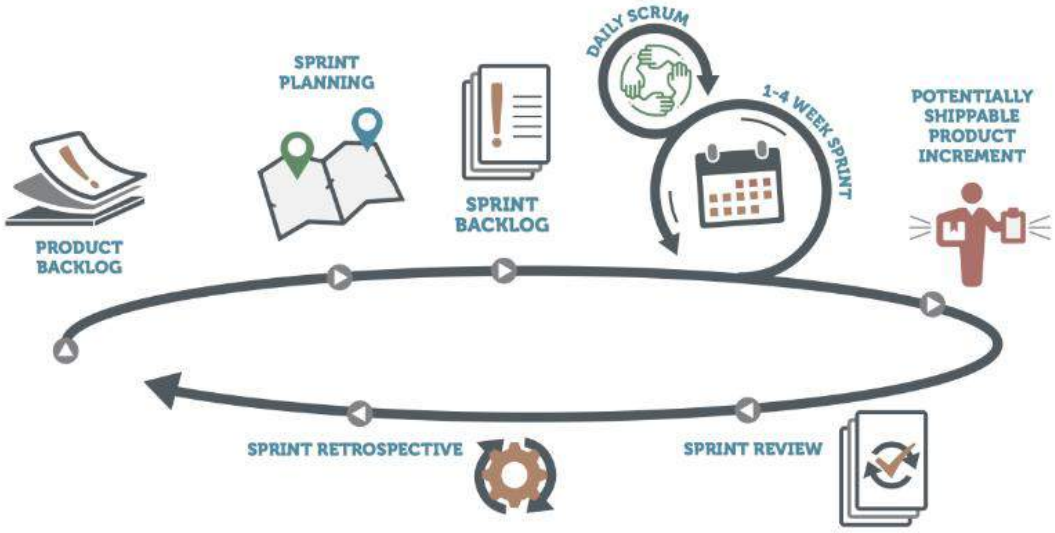
La metodología Scrum fue desarrollada en 1993 por Jeff Sutherland en asocio con Ken Schwaber cuando presentaron su idea a otros desarrolladores de productos de software en la conferencia Programación, Sistemas, Lenguajes y Aplicaciones Orientadas a Objetos (OOPSLA) de 1995. Schwaber y Sutherland continuaron combinando sus ideas en una metodología conjunta. Finalmente, Schwaber se asoció con Mike Beedle para escribir *Agile Software Development with Scrum*, lo que ayudó a convertir Scrum en un conjunto de herramientas más común en el movimiento de desarrollo ágil.

El eje principal del SCRUM es el *sprint*, durante el cual se crean incrementos de productos que pueden ser utilizables por los usuarios, es decir, entregas tempranas con valor. Los sprints suelen ser de una semana a un mes, y se generan secuencialmente con el fin de mantener el proyecto en marcha constante [18].

Todo el trabajo de diseño, desarrollo, prueba y validación del cliente está contenido en el sprint; el objetivo es que, al final del sprint, la nueva funcionalidad esté lista para entregarse a los usuarios [19].

La **Figura 1-1**, resume el proceso que se lleva a cabo en un proyecto implementado con metodología Scrum.

Figura 1-1: Metodología Scrum.



Fuente: agilealliance.org

Mientras los equipos de desarrollo llevan a cabo sus tareas, el proyecto debe ser monitoreado a través de reuniones diarias y revisar el estado de las tareas para mejorar las comunicaciones entre los equipos y mantener el enfoque en las metas del proyecto.

1.2.2.2 Kanban

El método Kanban fue desarrollado por David J. Anderson, quien estaba interesado en la entrega de software "justo a tiempo" y en la evolución del proceso para crear un sistema óptimo. Kanban se fundamenta en el *Signboard* que se describe como un sistema visual de gestión de procesos que les dice a los equipos qué producir, cuándo producirlo y cuánto producir [19].

Kanban se basa en cuatro principios fundamentales:

- Fomentar el liderazgo en todos los niveles de la organización: se motiva a todos en la organización a actuar como líderes.
- Comience con lo que sabe: Kanban no es de naturaleza prescriptiva y asume que las organizaciones no son todas iguales. En cambio, es importante comprender dónde se encuentra hoy y utilizar enfoques de gestión del cambio para evolucionar desde ese punto.
- Centrarse en el cambio incremental y evolutivo: Con frecuencia es importante hacer cambios pequeños pero que generen impacto.
- Respetar las metodologías y funciones actuales: es importante preservar lo que ha funcionado y cambiar lo que ya no ayuda al equipo a lograr sus objetivos.

Una diferencia importante de Kanban con respecto a otros enfoques es que no se basa en iteraciones. Kanban permite que el equipo se concentre al limitar el trabajo que está en progreso y aboga por un flujo continuo de trabajo. Kanban no prohíbe el uso de iteraciones o cuadros de tiempo, y es común que los equipos usen estas herramientas con Kanban, pero los marcos de tiempo específicos no se consideran un elemento necesario de este enfoque.

1.2.2.3 Extreme Programming

La idea del Extreme Programming (XP) comenzó en Chrysler Motors en 1996 con un proyecto de nómina denominado *Chrysler Comprehensive Compensation System* (C3). El equipo del proyecto, dirigido por Kent Beck, estaba luchando por cómo publicar código de alta calidad mucho más rápido y de manera más eficiente y, como resultado, transformó su proceso de desarrollo para que fuera eficiente, con un enfoque de calidad. En 1999, la popularidad de la programación extrema despegó cuando las ideas desarrolladas durante el proyecto C3 se documentaron en un libro escrito por Beck llamado *Extreme Programming Explained* [19].

Su nombre se debe a que los elementos de las prácticas tradicionales desarrollo de software se llevan a niveles “extremos”. Por ejemplo, las revisiones de códigos se consideran una práctica beneficiosa; llevada al extremo, el código se puede revisar continuamente, es decir, con la práctica de la programación por pares [20]. Extreme Programming se fundamenta en cinco valores principales que son: la comunicación, sencillez, realimentación, el coraje y el respeto.

Uno de los cambios que introdujo XP en el mundo de desarrollo de software, fue la idea de ciclos de desarrollo más pequeños y frecuentes. Los profesionales de XP argumentaron que el código es más útil que la documentación extensa y que los desarrolladores deben centrarse en desarrollar código de calidad en lugar de documentar lo que planean codificar [19].

1.2.2.4 Test Driven Development (TDD)

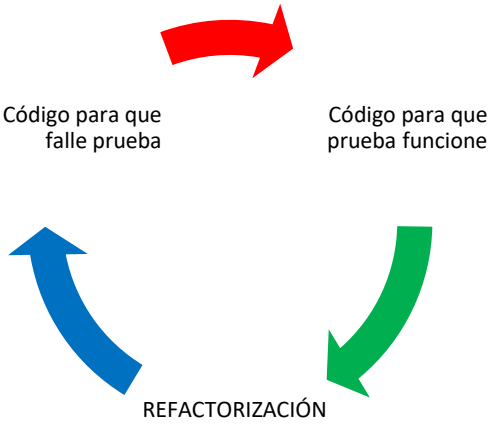
Si bien la idea de que la elaboración de pruebas preceda a la programación no es original de la comunidad Agile, TDD constituye un gran avance en la medida en que combina esa idea con la de "pruebas de desarrolladores”. TDD comienza en realidad a formarse en 1976 a partir de la publicación “Software Reliability” de Glenford Myers, que establece como idea clave que un desarrollador nunca debe probar su propio código.

Posteriormente, en 1990 se habla de disciplinas de prueba dominadas por técnicas de "caja negra", en particular en forma de herramientas de prueba de "captura y reproducción", además durante esta década de los 90 se da la creación del marco de pruebas SUnit y se escribe un artículo sobre Extreme Programming que menciona que "generalmente escribimos la prueba primero" y es así como surgen la relación entre metodologías ASD y la pruebas. Para 2006, TDD es una disciplina relativamente madura

que ha comenzado a fomentar nuevas innovaciones derivadas de ella, como ATDD o BDD [21].

El "desarrollo basado en pruebas" comprende tres actividades relacionadas entre sí: codificación, pruebas (se refiere a las pruebas unitarias realizadas por el desarrollador) y diseño (enfoque de refactorización) (**Figura 1-2**).

Figura 1-2: Ciclo Metodología TDD.



Fuente: Elaboración propia

Este ciclo, se repite tantas veces como sea necesario, bien sea dentro de un proyecto desarrollado únicamente con mitología TDD, o incorporando TDD como parte de un proyecto implementado con cualquier otra metodología ágil.

1.2.2.5 Lean Software Development (LSD)

El desarrollo de software Lean proviene del mundo de la fabricación, que ha adoptado ampliamente la idea de la fabricación Lean. LSD se basa en la idea de que todos los procesos y recursos deben contribuir directamente a crear algo de valor para el cliente. El valor para el cliente se define como cualquier cosa que un cliente está dispuesto a comprar. Cualquier cosa fuera de este alcance se considera un desperdicio y se debe hacer todo lo posible para eliminar el desperdicio. El desarrollo de software Lean se suscribe a los mismos principios, pero los relaciona con el desarrollo de software.

En 2003, Mary Poppendieck y Tom Poppendieck escribieron el primer libro sobre este tema, titulado Lean Software Development: An Agile Toolkit [22]. Su libro describe los siete principios que guían el desarrollo de software Lean que son: Elimine el desperdicio, amplifique el aprendizaje, pruebe el código con regularidad, decida lo más tarde posible, entregue lo más rápido posible, empodere al equipo, incorpore integridad, y vea el todo [19].

1.2.2.6 Adaptative Software Development

El desarrollo adaptativo de software es una metodología de desarrollo ASD cuya premisa es que el estado normal de las cosas es la adaptación continua del proceso al trabajo a desarrollar.

El desarrollo adaptativo de software reemplaza el ciclo tradicional en cascada con una serie repetida de ciclos de especulación, colaboración y aprendizaje. Este ciclo dinámico proporciona un aprendizaje continuo y una adaptación al estado emergente del proyecto. El proyecto implementado con Adaptative Development debe ser: centrado en el objetivo, basado en funciones, iterativo, fijo en el tiempo, guiado por riesgos y tolerante al cambio [20].

1.2.2.7 Crystal

Creada por Alistair Cockburn estableciendo que, cuando un equipo utiliza un proceso iterativo, la cantidad de estructura debe adaptarse a la dinámica del equipo. Cockburn usó el término "cristal" para la metodología porque no hay dos cristales exactamente iguales, en función de sus colores y dureza únicos; la analogía es que no hay dos proyectos iguales [19]. Esta metodología se diferencia de otras metodologías en que se centra en las personas por encima de los procesos o características específicos, pero comparte la premisa de otras metodologías ágiles al enfatizar la entrega frecuente a los usuarios, la conformación de equipos y un enfoque en la adaptación a través de revisiones periódicas [20].

La metodología Crystal se descompone en un grupo de métodos según el tamaño y complejidad como se describe a continuación [23].

- Crystal Clear: Para grupos de trabajo de hasta 6 personas o menos.

- Amarillo: Grupos de trabajo de 7 a 20 personas.
- Naranja: Grupos de trabajo de 21 a 40 personas.
- Roja: Grupos de trabajo de 41 a 80 personas.
- Marrón: Grupos de trabajo de 81 a 200 personas.

Algunos principios son comunes en todos los métodos Cristal. Uno de ellos es que los proyectos siempre usan ciclos de desarrollo con una duración máxima de cuatro meses, pero preferiblemente entre uno y tres meses. Además, el enfoque Crystal incorpora objetivos para reducir productos de trabajo intermedios y moldeando convenciones dentro de una metodología para proyectos individuales y desarrollándolos a medida que el proyecto evoluciona.

1.2.2.8 Feature Driven Development (FDD)

FDD es un método iterativo e incremental basado en dividir el software en diferentes modelos y construir cada modelo por separado. El proceso de desarrollo para cada modelo consta de cinco actividades: desarrollar el modelo general, construir la lista de características, planificar para desarrollar la característica, diseñar para la característica y construir por característica [24].

FDD se usa para proyectos grandes, porque se puede dividir en muchas tareas pequeñas, tareas que aumentan la posibilidad de completar el proyecto con éxito. Este también le da a la gerencia la posibilidad de cambiar el equipo mientras el proyecto está en ejecución esto sin afectar el cronograma del proyecto y la calidad general.

1.2.2.9 SAFe

El Scaled Agile Framework (SAFe), desarrollado por Dean Leffingwell, es una base de conocimiento interactiva para implementar prácticas ágiles a escala empresarial.
Con cinco actualizaciones importantes desde su lanzamiento inicial en 2011, SAFe ha crecido con el mercado a medida que se han ido descubriendo nuevas y mejores formas de desarrollar software [25].

SAFe se compone de tres niveles principales:

- Capa de Portafolio: la capa de portafolio es la capa donde los programas se alinean con la estrategia comercial y la intención de inversión de la empresa.
- Capa de programa: Scaled Agile Framework reconoce la necesidad de alinear e integrar los esfuerzos de múltiples equipos que participan en esfuerzos de desarrollo grandes y complejos a nivel empresarial.
- Capa de equipo: la capa de equipo forma la base de Scaled Agile Framework y es donde se realizan las actividades fundamentales de diseño y pruebas de construcción.

La última versión, SAFe 5, se basa en siete competencias básicas que son fundamentales para lograr y mantener una ventaja competitiva en una era cada vez más digital [26]. Estas son: avanzar y aplicar habilidades de liderazgo, impulsar comportamientos ágiles y técnicos del equipo, entrega ágil de las aplicaciones de software, entrega de soluciones empresariales, ejecutar la visión del portafolio y la formulación de la estrategia de financiación, agilidad organizacional para adaptarse rápidamente, y aumentar continuamente el conocimiento.

1.2.2.10 Dynamic Systems Development Method (DSDM)

El principio fundamental de DSDM se basa en la preferencia de fijar el tiempo y los recursos primero, y luego sí ajustar la cantidad de funcionalidad de acuerdo con esos tiempos y recursos establecidos inicialmente. [23]

DSDM consta de cinco fases: estudio de viabilidad, estudio de negocio, modelo de iteración funcional, iteración de diseño y construcción, e implementación.
Las primeras dos fases son secuenciales y se realizan una sola vez. Las tres últimas fases, durante las que se realiza el trabajo de desarrollo real, son iterativas e incrementales.
DSDM aborda las iteraciones como cajas de tiempo. Un *timebox* tiene una duración predefinida de tiempo y la iteración tiene que terminar dentro de ese tiempo establecido. El tiempo considerado para cada iteración a realizar se planifica con anticipación, junto con los resultados que es espera garantizar en esa iteración. En DSDM, una duración típica del timebox puede ser desde unos pocos días hasta algunas semanas.
Nueve prácticas definen la ideología y la base de toda actividad en DSDM. Las prácticas, llamadas principios en DSDM [23], se enumeran en la tabla que se presenta en la siguiente sección y que resume los principios de todas las metodologías ágiles descritas anteriormente.

1.2.2.11 Lista de principios en metodologías ASD

La **Tabla 1-1** resume los principios o prácticas fundamentales que se consideran dentro cada metodología ágil descrita anteriormente y que se contemplaron en este estudio.
De igual forma, se listan los roles sugeridos dentro de cada metodología en función del cumplimiento de dichos principios.

Tabla 1-1: Principios y roles en metodologías ágiles

Metodología ágil	Principios / practicas fundamentales	Roles
SCRUM	-Organizar proyectos en duraciones específicas. -Alto trabajo en equipo. -Sprints de 1 a 4 semanas con incremento de productos. -Reuniones diarias. -Historias de usuario.	-Product Owner -Scrum master -Equipo de desarrollo
KANBAN	-Visualizar el flujo de trabajo. -Limitar el trabajo en progreso. -Administrar el flujo. -Proporcionar políticas explícitas. -Colaboración y evolución. -Realimentación constante.	No obligatorios -Administrador de solicitudes de servicio -Gerente de Servicio de Entregas
Extreme Programming-XP2	-Priorización de historias de usuario. -Ciclos de 1 a 3 semanas. -Diseño incremental. -Programación de prueba primero. -Integración continua. -Historias de usuario	-Desarrollador -Cliente -Tester -Tracker -Coach -Gestor
Test Driven Development (TDD)	-Escribir código de una funcionalidad cuya prueba falle. -Ejecutar la prueba que debería fallar. -Escribir el código para que la prueba funcione. -Refactorizar el código hasta que se ajuste. -Repetir pruebas unitarias a lo largo del tiempo.	-Cliente -Analista de negocio -Desarrolladores -Arquitectos -Administradores de Sistemas
Lean Software Development (LSD)	-Eliminar el desperdicio. -Amplificar el aprendizaje. -Decidir lo más tarde posible. -Entregar lo más rápido posible. -Empoderar al quipo. -Incorporar integridad y ver el todo.	No específicos
Adaptative Software Development (ASD)	-Centrado en el objetivo. -Basado en componentes. -Iterativo. -Fijo en el tiempo. -Tolerancia a riesgos. -Tolerancia a los cambios.	No específicos

Tabla 1-1: (Continuación)

Metodología ágil	Principios / practicas fundamentales	Roles
Crystal	-Se centra en las personas por encima de los procesos. -Enfatiza la entrega frecuente a los usuarios. -Ciclos de 1 a 3 meses -Conformación de equipos. -Adaptación a través de revisiones periódicas.	-Patrocinador -Diseñador-programador senior, -Diseñador-programador -Usuario
Feature Driven Development (FDD)	-Descripción el modelo general. -Creación de lista de características (funcionalidades). -Planificar la característica. -Diseñar de la característica. -Construir por característica.	Claves/soporte/adicionales -Administrador del proyecto -Arquitecto jefe -Manager de desarrollo -Programador jefe -Propietarios de clases -Experto de dominio
SAFe	-Liderazgo. -Agilidad técnica y de equipo. -Entrega ágil de productos. -Entrega de soluciones empresariales. -Ejecutar visión y estrategia. -Agilidad organizacional. -Cultura de aprendizaje continuo.	-Release Train Engineer -Product Magnament -System Architect -Bussiness Owner
Dynamic Systems Development Method (DSDM)	-La participación del usuario es imperativa. -Frecuentes entregas de productos. -El desarrollo iterativo e incremental. -Todos los cambios durante el desarrollo son reversibles. -Requisitos de alto nivel. -Enfoque cooperativo. -Pruebas durante todo el ciclo.	No específicos

1.3 Estimación de esfuerzo

La estimación es el acto de determinar un valor probable de una variable. Es usualmente aplicado a proyectos para pronosticar su costo, duración, el esfuerzo y los recursos requeridos [27]. La estimación del esfuerzo de desarrollo de software (SDEE, Software Development Effort Estimation) se define como el proceso de predecir el esfuerzo requerido para construir una pieza de software [28]. La unidad de esfuerzo es la hora-hombre o el mes-hombre [3].

En el dominio de la ingeniería de software, la estimación del esfuerzo y la planificación son dos actividades estrechamente relacionadas desde el principio; la primera es imprescindible para llevar a cabo la segunda [29] .

En un proyecto de software, las estimaciones exactas y precisas influyen en varias tareas de planificación que ayudan a completar el proyecto correctamente, incluida la creación de un cronograma detallado, dividir un proyecto en iteraciones y priorizar la funcionalidad para la entrega.

Los métodos de estimación de esfuerzo en ingeniería del software cubren una amplia gama de enfoques que se utilizan además como insumo para estimar el tiempo y costo del software a desarrollar.

1.3.1 Métodos para estimación de esfuerzo

Los investigadores de software han ido desarrollando numerosos métodos de estimación del esfuerzo. Cada nuevo método propuesto es más avanzado y pretende ser el que resuelve problemas que no han sido abordados antes. Sin embargo, sólo algunos de los métodos han sido aceptados en la industria del software [29]. Estos se agrupan con respecto a dos aspectos básicos: (1) el tipo de datos que utilizan como entrada y (2) el principio de estimación que emplean. Una tercera opción, es la combinación de ambos, que se conocen como híbridos.

Basados en datos: La estimación del esfuerzo basada en datos se refiere a métodos que predicen el esfuerzo basándose únicamente en el análisis cuantitativo de los datos históricos de los proyectos. En el curso de la estimación, las relaciones entre el esfuerzo real del proyecto y las características del proyecto son exploradas con base a los datos de medición recopilados de proyectos ya completados [29].

Las relaciones encontradas se proyectan luego en un nuevo proyecto para predecir el esfuerzo esperado.

Basados en Juicio de Expertos: La estimación basada en el juicio de expertos sigue siendo el método de estimación más popular en la industria del software. Por lo general, entre el 70 y el 80 % de las estimaciones las realizan expertos sin utilizar modelos formales de estimación [29]. Los métodos basados en expertos implican consultar con uno o más expertos, quienes usan su experiencia en el campo, así como la comprensión del contexto de la organización para estimar el costo del proyecto.

Híbridos: Todos y cada uno de los métodos de estimación tienen sus puntos fuertes y sus limitaciones, y su bondad depende en gran medida del contexto en el que se aplica. Los métodos híbridos comprenden una combinación de múltiples paradigmas de estimación, como métodos basados en expertos y basados en datos. En un método híbrido se combinan estimaciones múltiples proporcionadas por métodos de estimación independientes, preferiblemente representando diferentes paradigmas de estimación.
A la luz de la falta de consenso sobre qué enfoques basados en expertos o basados en datos son "mejores", los métodos híbridos ofrecen la combinación de las fortalezas de ambas estrategias evitando sus debilidades [29].

Aunque existen muchas técnicas de estimación, dentro de este campo todavía hay margen de mejora. Recientes estudios empíricos en estimación de esfuerzo ágil (AEE, Agile Effort Estimation) han utilizado algoritmos de Machine Learning (ML) para estimar el esfuerzo [30]. Estos modelos muestran mejoras en la exactitud y precisión para enfoques ágiles. Por otro lado, una actualización reciente de una Revisión Sistemática de Literatura (SLR) sobre la estimación del esfuerzo en ASD resume los resultados de estudios primarios publicados desde diciembre de 2013 hasta enero de 2020. La revisión se centra en 73 artículos que proporcionaron nueva evidencia para caracterizar las actividades de estimación en proyectos ágiles. Los resultados confirman que Planning Poker se ha convertido en el Marco de estimación más utilizado, y está muy relacionado con la métrica de tamaño más utilizada: Puntos de historia (SP, Story Points). Además, SP mide el tamaño y la complejidad de las historias de usuario, que son los principales artefactos para la especificación de los requisitos del usuario en ASD [6].
En general, la estimación puede verse como un conjunto de factores del proyecto que pueden combinarse de alguna forma para proporcionar la estimación del esfuerzo.

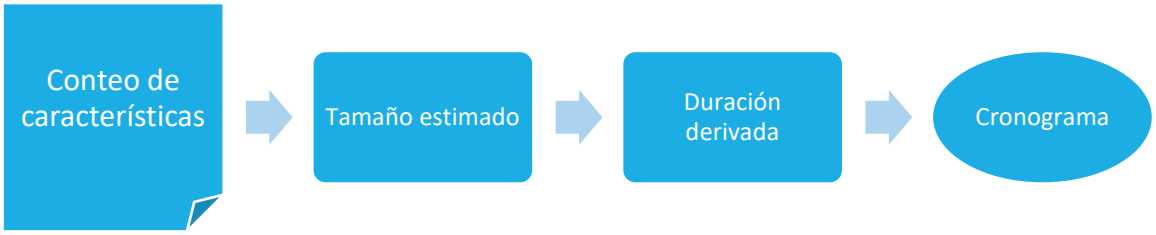
1.3.1.1 Unidades de medida

Los equipos ágiles separan las estimaciones de tamaño de las estimaciones de duración [31]. Al estimar el esfuerzo del proyecto de software, debe haber algunas entradas que describan los requisitos del proyecto. La precisión y la integridad de estas entradas son un problema significativo. Debido a que las entradas en sí mismas son en su mayoría estimaciones, se hace necesario convertirlas en un solo número expresado en alguna unidad de medida, como personas-meses [32]. Una vez expresado el esfuerzo en algunas

unidades, se debe afrontar el problema de la uniformidad. En otras palabras, una persona-mes puede variar drásticamente según el nivel de habilidad de la persona asignada. A pesar de estos problemas, todavía es necesario estimar el esfuerzo del proyecto y construir un plan. Con tanta incertidumbre, no es difícil ver por qué las fases de seguimiento y ajuste son cruciales para la gestión de proyectos.

El objetivo final de cada estándar de estimación de esfuerzo es llegar a una medida de tamaño en función de una unidad que pueda ser traducida finalmente en horas de trabajo teniendo en cuenta factores como la velocidad del equipo y los recursos disponibles. Todo esto será finalmente el insumo para la construcción de un cronograma. Esto lo describe Mike Cohn [31] de un modo muy práctico como se muestra la **Figura 1-3**.

Figura 1-3: Estimación de la duración de un proyecto.



Fuente: Elaboración propia

La estimación de la duración de un proyecto comienza con la estimación de su tamaño.

Referente a la estimación del tamaño, el recuento de líneas de código para un proyecto suele ser un gran indicador, pero el código no estará disponible para contar hasta el final del proyecto. Los puntos de función también podrían ser considerados y están fuertemente asociados con el tamaño final del proyecto, pero no están disponibles hasta que se tengan requisitos detallados.

Si se puede encontrar algo que se pueda contar antes para tener una aproximación del tamaño, se puede usar para crear una estimación de esfuerzo antes. Por ejemplo, se puede crear una estimación aproximada basada en un recuento de los requisitos y luego ajustar la estimación más tarde en función de un recuento de puntos de función [33].

Se describen a continuación de forma breve, algunas unidades de medida para estimación esfuerzo que serán señaladas más adelante en la definición de los estándares de estimación (**Tabla 1-2**).

Tabla 1-2: Unidades de medida

Story Points	Story Points (SP) son una medida para estimar el nivel de esfuerzo asociado con la implementación de una historia de usuario. La forma típica de los puntos de la historia es una serie de Fibonacci como 1, 2, 3, 5, 8, 13, siendo 1 un nivel mínimo de esfuerzo y 13 un nivel máximo de esfuerzo para una historia de usuario en una iteración.
Function Points	Los puntos de función se usan para cuantificar la funcionalidad del software a desarrollar. Esto se hace identificando y clasificando componentes como las entradas y salidas externas, interacciones de usuario, interfaces externas y archivos o tablas de bases de datos que utiliza el sistema. Una vez que dichos componentes han sido identificados y clasificados, se evalúan por su complejidad funcional utilizando un conjunto de atributos prescritos. Las funciones se clasifican en complejidad baja, media o alta. Los componentes funcionales reciben Puntos de Función según su clasificación de tipo y categorización de complejidad.
Líneas de código	Consiste básicamente en realizar un estimado de las líneas de código que serían necesarias para cumplir con una funcionalidad específica. De igual forma, en un análisis realizado por ejemplo por analogía, este ejercicio consistiría en contar líneas de código de un desarrollo anterior y con base en esto estimar el nuevo desarrollo.
Use Case Points	El concepto y uso de UCP es similar al de los FP. La estimación con UCP requiere que todos los casos de uso (documentos) se escriban con un objetivo y aproximadamente al mismo nivel, brindando la misma cantidad de detalles. Esto quiere decir que, antes de realizar la estimación, se debe contar con todos los casos de uso documentados de forma detallada.
Object Points	Los puntos de objeto son un enfoque utilizado en la estimación del esfuerzo de desarrollo de software en algunos modelos como COCOMO II. Los puntos de objeto son una forma de estimar cantidad de esfuerzo, similar a las líneas de código fuente o los puntos de función. Los objetos a los que se hace referencia incluyen pantallas, informes y módulos del lenguaje. Se calcula la cantidad de objetos sin procesar y la complejidad de cada uno, y luego se calcula un conteo total ponderado de puntos de objetos y se usa para basar las estimaciones del esfuerzo necesario.
Work hours	Medida del esfuerzo que se refiere a la cantidad de tiempo en horas, requerido por un desarrollador para implementar determinada funcionalidad de un software. Para esto, el primer paso debe ser obtener una buena comprensión de cada tarea que debe realizarse. El segundo paso es hacer suposiciones razonables para el proceso detrás de cada tarea, así como las condiciones bajo las cuales se realiza una tarea. Por última, el experto estima el tiempo en horas requerido para llevar a cabo el desarrollo.

1.3.1.2 Juicio de Expertos – Delphi

El juicio de expertos se define como el juicio apropiado basado en la experiencia de una aplicación, área, campo de conocimiento, disciplina o industria, para la actividad que se realiza. Dicha experiencia puede ser aportada por grupos o individuos con educación, conocimientos, habilidades, experiencia o capacitación específicos [2].

Las técnicas de juicio de expertos son el enfoque de estimación más mencionado dentro de la Guía PMBOK para predecir el esfuerzo de desarrollo software.
Se sugiere el juicio de expertos como una herramienta/técnica potencial en los seis procesos contenidos en el Área de conocimiento de gestión de integración de proyectos. En el otro extremo del espectro, el juicio de expertos no se menciona explícitamente como una herramienta/técnica para ninguna de las tres áreas de conocimiento de gestión de la calidad del proyecto.

El juicio de expertos Delphi, implica el llegar a un consenso basado en la discusión entre expertos mediante un proceso iterativo, es decir, el juicio debe ser sometido a discusión grupal. El método Delphi, implica la elaboración de un cuestionario que debe ser respondido por los expertos en la materia a estimar. Una vez analizados estos resultados a nivel global, se debe diseñar otro cuestionario para ser respondido nuevamente por los expertos. Este proceso se repetirá tantas veces como se quiera hasta lograr un consenso. Lo anterior quiere decir que el juicio de expertos es en sí una técnica cualitativa que puede ser lleva a cabo por un solo individuo o por grupos de expertos (Delphi)

Esta técnica se considera una de las técnicas más precisas [33], pero su ejecución lleva tiempo y la necesidad de la participación de los expertos, por lo que no siempre es aplicable en grupo y puede estar sujeta a la participación de una sola persona que en muchos casos será incluso quién desarrollará la funcionalidad del software objetivo dentro del proyecto.

1.3.1.3 Planning Poker

El marco de estimación más utilizado en ASD es Planning Poker, que se clasifica como una técnica Delphi. Es una variación del Juicio de Expertos en la que un grupo de expertos llega a un consenso sobre una estimación, después de realizar una o más rondas de

discusión y estimaciones anónimas individuales. La estimación individual realizada en cada ronda es simultánea y anónima para eliminar sesgos [27].

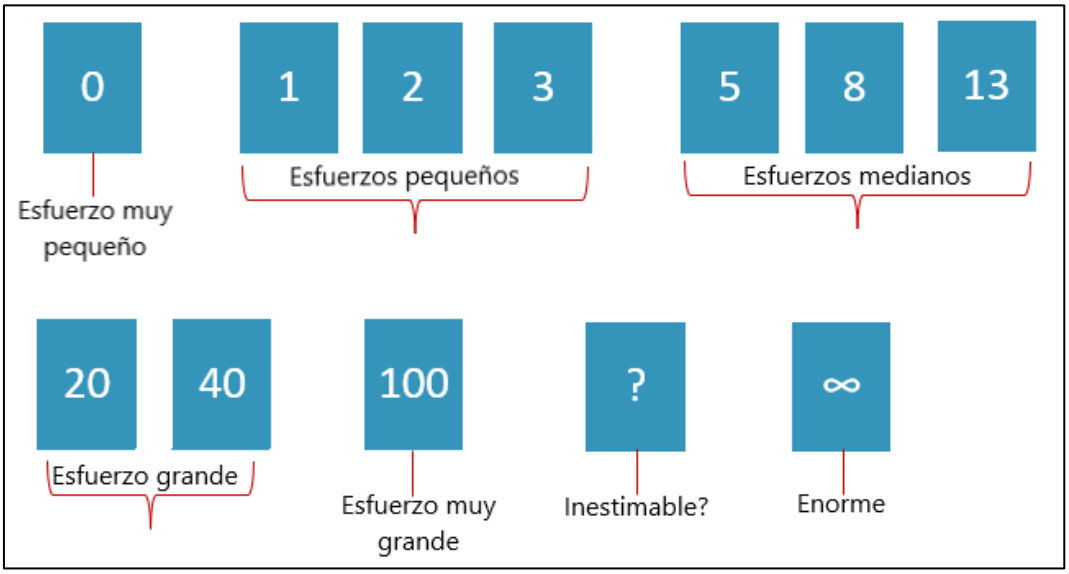
Planning Poker combina la opinión de expertos, la analogía y la desagregación para obtener estimaciones rápidas y confiables. Los participantes en la planificación incluyen a todos los desarrolladores del equipo. El propietario del producto (Product Owner) participa en la planificación, pero no hace estimaciones [31].

Al comienzo de la planificación, cada estimador recibe una baraja de cartas con la secuencia de Fibonacci modificada (pseudo Fibonacci). Cada tarjeta tiene escrito en ella una de las estimaciones válidas. Cada estimador puede, por ejemplo, recibir una baraja de cartas que diga 0, 1, 2, 3, 5, 8, 13, 20, 40 y 100. Las cartas deben prepararse antes de la reunión de planificación y los números deben ser lo suficientemente grandes para ver a través de una tabla. Las tarjetas se pueden guardar y usar para la próxima sesión de planificación.

Para cada historia de usuario a estimar, un moderador lee la descripción. El moderador suele ser el propietario del producto o un analista del negocio, sin embargo, puede ser cualquier persona. El dueño del producto responde cualquier pregunta que tengan los estimadores. Después de responder todas las preguntas, cada estimador selecciona en privado una tarjeta que representa su estimación. Las tarjetas no se muestran hasta que cada estimador haya hecho una selección. En ese momento, todas las tarjetas se dan vuelta y se muestran simultáneamente para que todos los participantes pueden ver cada estimación.

La utilidad de hacer uso de la secuencia de Fibonacci se debe a que la diferencia entre los valores de dicha secuencia sirve para reflejar la incertidumbre en la estimación de elementos que pueden ser grandes y para expresar los diferentes niveles de dificultad de una tarea. Dado que Planning Poker es finalmente una técnica basada en juicio de expertos, las cartas que representan los números de la serie de Fibonacci se pueden considerar como se muestra **Figura 1-4** para mejor orientación de los estimadores.

Figura 1-4: Esfuerzo con Planning Poker.



Fuente: Elaboración propia

En muchos casos, las estimaciones ya convergerán en la segunda ronda. Pero si no es así, se repite el proceso. El objetivo es que los estimadores lleguen a una sola estimación que pueda usarse para la historia de usuario.

1.3.1.4 Analogia

El método de analogía utiliza la experiencia de proyectos anteriores. Este método compara el proyecto propuesto con proyectos similares previamente completados donde se conoce la información real del desarrollo del proyecto. Un estimador experimentado puede producir estimaciones de tamaño razonablemente buenas por analogía si se dispone de valores de tamaño precisos para el proyecto anterior y si el nuevo proyecto es lo suficientemente similar al anterior [34].

Este proceso básico de estimación por analogía puede producir mejores resultados [33]:

- Obtener datos detallados de tamaño, esfuerzo y costo de un proyecto anterior similar. Si es posible, obtener la información por categoría de estructura de descomposición del trabajo (EDT) o por algún otro esquema de descomposición.

- Comparar el tamaño del nuevo proyecto pieza por pieza con el proyecto anterior.
- Construir la estimación del tamaño del nuevo proyecto como un porcentaje del tamaño del proyecto anterior.
- Crear una estimación de esfuerzo basada en el tamaño del nuevo proyecto en comparación con el tamaño del proyecto anterior.
- Comprobar si hay suposiciones en común entre los proyectos antiguos y nuevos.

1.3.1.5 Top-Down

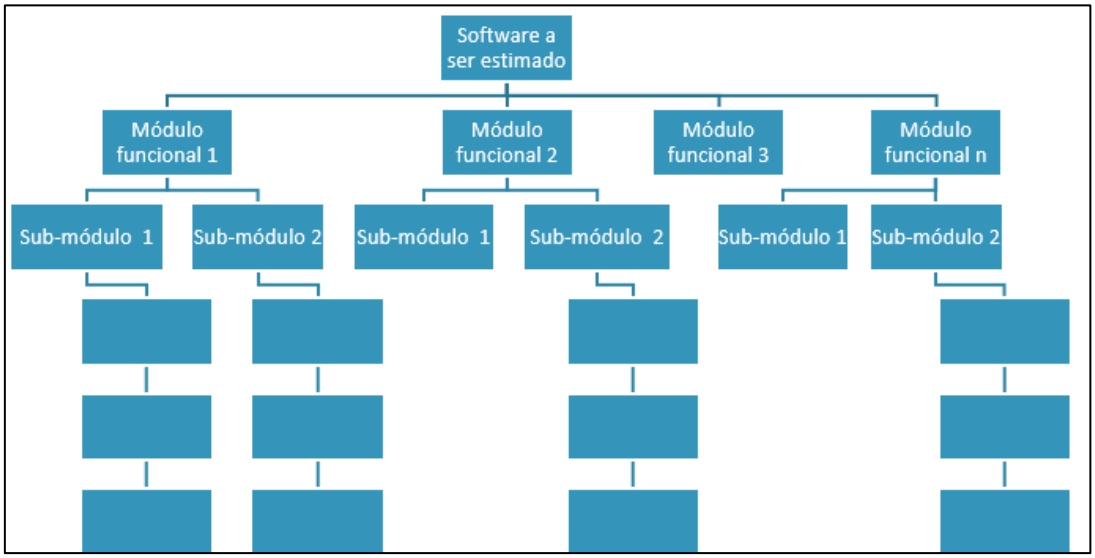
Este método sigue el orden de arriba hacia abajo de trabajar con los módulos principales, los submódulos y las funciones individuales. Este método también se conoce como el método de estructura de desglose del trabajo (Work-Breakdown Structure -WBS). Existen otras alternativas a este método que incluyen el desglose de actividades por ciclo de vida y la estimación a nivel de actividad [34].

Las mejores técnicas para grandes proyectos cambian significativamente desde el comienzo del proyecto hasta el final. En las primeras etapas, los mejores enfoques de estimación tienden a ser técnicas "Top-Down" basadas en algoritmos y estadísticas. Estos son válidos en el punto del proyecto en el que aún no se conocen los miembros específicos del equipo, cuando los planes se basan en un equipo que consta de una cantidad de profesionales con roles específicos, en lugar de personas específicas.

El método WBS ofrece una serie de enfoques alternativos para definir la estructura de arriba hacia abajo del sistema que se estima. Dos posibles formas son:

- Centrado en el producto: la aplicación se divide en el producto principal, subproducto, módulos, elementos y componentes. El esfuerzo o costo real estimado para cada componente se evalúa y luego se agrega hacia arriba, **Figura 1-5**.
- Centrado en el ciclo de vida del proyecto: las actividades que están involucradas en la ejecución del proyecto se desglosan por ciclo de vida (por ejemplo, requisitos, diseño, construcción y prueba). Cada fase del ciclo de vida se divide en actividades individuales. Se hacen estimaciones para actividades individuales y luego se agregan hacia arriba.

Figura 1-5: Descomposición del trabajo - Top-Down.



Fuente: Elaboración propia

En las etapas intermedias, una combinación de técnicas Top-Down y Botton-Up basadas en los propios datos históricos del proyecto producirá las estimaciones más precisas. En las últimas etapas de grandes proyectos, las técnicas de Botton-Up proporcionarán las estimaciones más precisas.

1.3.1.6 Bottom-Up

A diferencia del enfoque Top-Down, donde los datos sobre la aplicación, sus módulos y programas no se conocen detalladamente, el enfoque Bottom-Up requiere que la información sobre los componentes se estime al nivel más granular. Esto significa que es esencial que se complete una fase detallada de recopilación y análisis de requisitos antes de intentar el enfoque de estimación Bottom-Up.

Este método adopta el enfoque ascendente al identificar primero todos y cada uno de los componentes o actividades a un nivel muy específico. Luego, el método estima por separado cada componente y posteriormente consolida los resultados para producir una estimación de todo el proyecto [34].

En este enfoque, el trabajo del proyecto se divide primero en módulos principales. Cada módulo se divide a su vez en programas. Los programas se clasifican además como simples, medios o complejos y el esfuerzo estimado para construir cada programa se basa en la experiencia pasada de proyectos similares. Este método estima cada componente del proyecto de software por separado y luego combina los resultados para producir una estimación de todo el proyecto. La estimación se realiza cuando los requisitos son claros o han sido aprobados.

El método de enfoque Bottom-Up tiene ciertas ventajas:

- Proporciona una base más detallada y precisa para la estimación, porque trata con componentes de bajo nivel.
- Admite el seguimiento de proyectos de manera más directa que otros métodos porque sus estimaciones generalmente abordan cada actividad dentro de cada fase del ciclo de vida del desarrollo de software.

Algunas desventajas conocidas de este método son que las estimaciones de abajo hacia arriba se pueden hacer solo después de que se hayan completado los requisitos y las fases de diseño, al igual que estimar los componentes a nivel granular consume mucho tiempo. La estimación Bottom-Up es el complemento de la estimación de Top-Down, en la que los ejecutantes responsables realizan las estimaciones de los componentes y luego se consolidan en una estimación de costos del sistema completo. Sus fortalezas complementan las debilidades de la estimación de Top-Down, como incorporar el compromiso de los ejecutantes para cumplir con sus estimaciones y basarse en un conocimiento detallado del trabajo a realizar. Sus debilidades, como centrarse en los costos de los componentes y los costos potencialmente faltantes de todo el sistema, como la integración del sistema, la gestión de la configuración y la gestión de proyectos, se complementan con las fortalezas de la estimación de Top-Down [35].

1.3.1.7 Regresión

El análisis de regresión es una herramienta estadística para la investigación de relaciones entre variables. Por lo general, el investigador busca determinar el efecto causal de una variable sobre otra. Para explorar estos temas, el investigador reúne datos sobre las variables subyacentes de interés y emplea la regresión para estimar el efecto cuantitativo

de las variables causales sobre la variable en la que influyen. El investigador también suele evaluar la "significación estadística" de las relaciones estimadas, es decir, el grado de confianza de que la verdadera relación es cercana a la relación estimada. El análisis de regresión se aplica para la predicción de la confiabilidad del software [36].

En el contexto de la estimación del esfuerzo, la variable dependiente es el esfuerzo y las variables independientes son los factores del esfuerzo; es decir, las características del proyecto de software que tienen un impacto en el esfuerzo. Los factores de esfuerzo incluyen el tamaño y la complejidad del software, las capacidades del equipo de desarrollo o la volatilidad de los requisitos.

La variable dependiente también se conoce comúnmente como una respuesta o variable endógena, mientras que las variables independientes también se denominan variables explicativas o exógenas.

En general, un modelo de regresión lineal simple tiene la forma de una ecuación, como se presenta a continuación

(1. 1)

$$Y = \alpha + \beta \times X + \varepsilon \qquad (1.\ 1)$$

La Y representa la variable dependiente y la X representa la variable independiente. El coeficiente β se denomina coeficiente de regresión, mientras que α es el intercepto o constante. Finalmente, el término ε es una variable aleatoria con media cero que representa el error que refleja efectos aleatorios causados por factores no observados y no explicados por la variable explicativa considerada en el modelo de regresión. El término ε también es comúnmente conocido como un término de "ruido" y se supone que sigue una distribución normal con media 0. Los valores reales de Y (el esfuerzo del proyecto observado) se supone que son la suma del valor medio μ (Y) y el término de error ε.

Gráficamente, si se ignora el término de error ε, el modelo de regresión representa una línea con una “intersección” de α en el eje vertical y una “pendiente” de β.

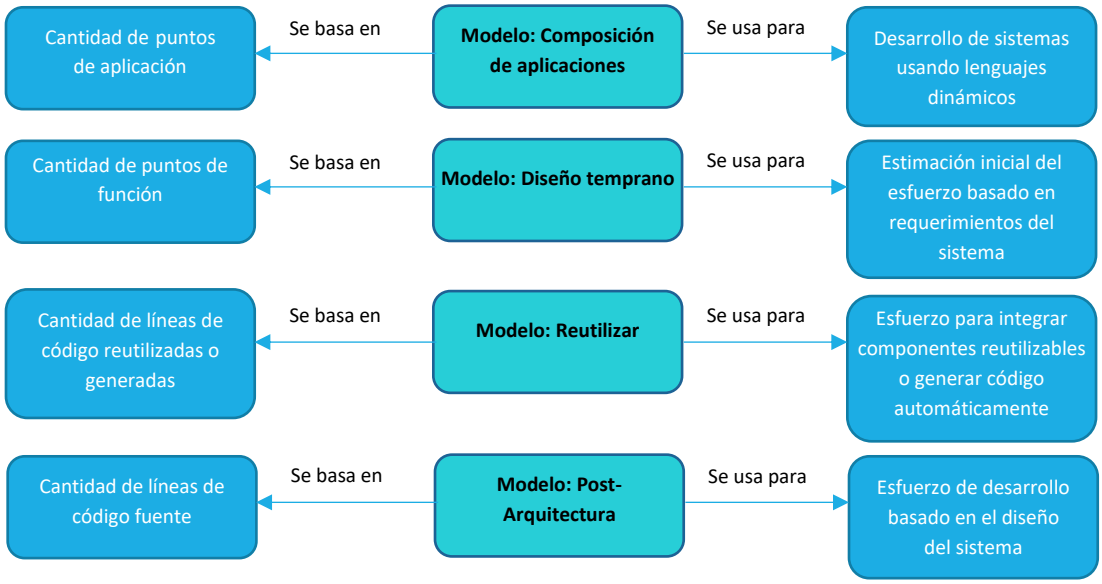
La línea de regresión no se conoce en la práctica y se aproxima estimando los coeficientes de regresión sobre una muestra de datos de observación utilizando una de las muchas existentes técnicas de análisis de regresión [29].

1.3.1.8 Constructive Cost Model - COCOMO II

COCOMO II corresponde a un modelo posterior a los modelos iniciales de estimación de costos COCOMO (Modelado de costos constructivos). El modelo COCOMO II considera enfoques modernos para el desarrollo de software, como el desarrollo rápido utilizando lenguajes dinámicos, el desarrollo con reutilización y la programación de bases de datos. COCOMO II incorpora varios submodelos basados en estas técnicas, que producen estimaciones cada vez más detalladas [3].

Los submodelos (**Figura 1-6**) que forman parte del modelo COCOMO II son:

Figura 1-6: Modelos de estimación COCOMOII.



Fuente: Adaptado de **[3]**

- Modelo de composición de aplicaciones: Determina el esfuerzo necesario para crear o desarrollar software a partir de componentes reutilizables, secuencias de comandos o programación de bases de datos. Las estimaciones del tamaño del software se basan en puntos de aplicación y se utiliza una fórmula simple de tamaño/productividad para estimar el esfuerzo requerido.
- Modelo de diseño temprano: Las estimaciones se basan en puntos de función (Function Points), que luego se convierten en número de líneas de código fuente. Este

modelo se utiliza durante las primeras etapas del diseño del sistema después de que se hayan establecido los requisitos.

- Modelo de reutilización: Este modelo se utiliza para calcular el esfuerzo necesario para integrar componentes reutilizables y/o código de programa generado automáticamente. Normalmente se utiliza junto con el modelo de post-arquitectura.
- Modelo post-arquitectura: Una vez que se ha diseñado la arquitectura del sistema, se puede realizar una estimación más precisa del tamaño del software. En este modelo se usa una fórmula estándar para la estimación de costos que también se utiliza en el modelo de diseño temprano. Sin embargo, para el modelo post-arquitectura se incluye un conjunto más amplio de 17 multiplicadores que reflejan la capacidad del personal, el producto y las características del proyecto.

El uso de COCOMO tiene como ventaja que los datos de muchos programas reales pueden proporcionar un conjunto de constantes COCOMO y factores de ajuste que se adaptan bien a una organización. Por otro lado, es un proceso repetible, el método permite la adición de factores de ajuste únicos asociados con una organización y es lo suficientemente versátil como para admitir diferentes "modos" y "niveles", está minuciosamente documentado y es fácil de usar [37].

1.3.1.9 Orientados al aprendizaje - Machine Learning (ML)

Machine Learning ha atraído a investigadores mientras diseñan modelos de software para la estimación de costos y esfuerzos. Según [38], estos modelos pueden diseñarse usando técnicas de ML individualmente, o usando técnicas en conjunto con otros enfoques de ML. Se conocen ocho tipos de técnicas orientadas al aprendizaje:

- Case-Based Reasoning (CBR)
- Artificial Neural Networks (ANN)
- Decision Trees (DT)
- Bayesian Networks (BN)
- Support Vector Regression (SVR)
- Genetic Algorithms (GA)
- Genetic Programming (GP)
- Association Rules (AR)

De esta lista, las técnicas CBR, ANN y DT son las más frecuentemente utilizadas. Estas se relacionan en el 80% de los estudios consultados en [28]. Por otro lado, [39] señala

como principales técnicas de ML utilizadas para estimación de esfuerzo las siguientes: Artificial Neural Network (ANN), Fuzzy Logic, y Genetic Algorithms (GA).
Para comprender mejor la relación entre ML y la estimación de esfuerzo, se describen brevemente a continuación las tres técnicas más señaladas en la literatura consultada.

ANN: Una ANN es la interconexión de un número de neuronas que imita el comportamiento de las neuronas biológicas. La organización de las neuronas en ANN se denomina topología de ANN. Las neuronas están organizadas a través de capas y conectadas a través de líneas que representan pesos. Se utiliza para predecir los esfuerzos de desarrollo de software en función de diferentes multiplicadores de esfuerzo, como el tamaño del software, la confiabilidad, la complejidad, la cantidad de puntos de función, la capacidad del analista, la capacidad del desarrollador y muchos más. Los multiplicadores de esfuerzo y los esfuerzos de desarrollo se consideran como entrada y salida de ANN, respectivamente. La relación no lineal entre la entrada y la salida de ANN se obtiene entrenando ANN. La red de alimentación directa es el tipo más simple de ANN utilizado para problemas de predicción y clasificación [36].

CBR*:* El razonamiento basado en casos (CBR) es similar al procesamiento analógico. En CBR, un problema (objetivo) se resuelve siguiendo los siguientes pasos: (1) se recupera un caso, o experiencia pasada, similar al problema actual; (2) el caso se utiliza para proponer una solución; (3) se evalúa la solución; y (4) se reparan los problemas con la solución. Debido a que CBR se basa en un solo caso o en un pequeño número de casos, CBR puede tener dificultades con problemas cuya solución requiere la combinación de muchos casos [40] .

Fuzzy Logic: La lógica difusa fue introducida originalmente como una forma matemática de representar la vaguedad en la vida cotidiana. La lógica difusa, como su nombre indica, es definida como la lógica subyacente a los modos de razonamiento que son aproximados en lugar de exactos [36]. La importancia de la lógica difusa se deriva del hecho de que la mayoría de los modos de razonamiento humano, y especialmente el razonamiento de sentido común, son de naturaleza aproximada. Es por todas estas razones que se considera que los sistemas de inferencia de lógica difusa son ideales para capturar información subjetiva de métricas de software durante la fase inicial de desarrollo de software. La mayoría de las primeras fases de las métricas del software no son muy claras

para el desarrollador y es difícil asignar un tema claro. Por lo tanto, se encontró que la lógica difusa es una herramienta adecuada para tratar con ellos.

1.3.1.10 Juicio de Expertos - Delphi de banda ancha

El método Wideband es una forma estructurada de estimación basada en la experiencia colectiva. Muy a menudo, el método Delphi de banda ancha se usa para realizar una validación cruzada de las estimaciones que se han realizado con otros métodos de estimación populares. Esto se basa en el reconocimiento del hecho de que cuando muchos expertos llegan de forma independiente a la misma estimación basándose en los mismos supuestos, es probable que la estimación sea correcta. “El enfoque de consenso ayuda a eliminar el sesgo en las estimaciones producidas por expertos autoproclamados, estimadores sin experiencia o personas influyentes que tienen intereses ocultos” [34].

Según [41] Durante la implementación del método Delphi de banda ancha se consideran las siguientes actividades:

- Se identifica un conjunto de profesionales de software experimentados que han participado en actividades relacionadas con la estimación en el pasado.
- Se proporciona una copia de los requisitos a todos los estimadores. También se proporciona un formulario bien estructurado que puede capturar detalles de la estimación y otros atributos relacionados.
- Se organiza una reunión preliminar donde todos los estimadores pueden reunirse, discutir e intercambiar puntos de vista, suposiciones y otros parámetros que son aplicables al proyecto que se está estimando.
- Los estimadores elaboran una lista de tareas y otros artefactos que probablemente se generarán durante la ejecución del proyecto. Junto con cada tarea, incluida la documentación si la hay, se registra el esfuerzo estimado. Los datos se introducen en el formulario habilitado al efecto. Luego se entrega el formulario completo al moderador.
- El moderador tabula los datos de todos los formularios provistos por los estimadores, los analiza, genera los resultados y los entrega a los expertos.

- Los expertos se reúnen nuevamente para discutir los resultados. En la reunión, los expertos revisan las tareas que han considerado para la estimación, refinan sus estimaciones individuales y entregan los datos revisados al moderador. Esto se repite hasta que haya convergencia en las estimaciones de los diferentes estimadores.

Por lo general, la convergencia ocurre después de dos o tres iteraciones. La decisión de pasar por más iteraciones se toma en función de la brecha entre las estimaciones más baja y alta. Una vez que la brecha se reduce a un nivel cómodo para todos los estimadores, el grupo llega a una cifra de consenso para el valor final.

1.3.2 Medición de exactitud

La práctica actual común al comparar modelos de estimación o al evaluar el desempeño de las organizaciones respecto a la estimación de software es aplicar las medidas de precisión: Magnitud Media de Error Relativo (MMRE) y Percentage Relative Error Deviation (PRED) o similares [42].

En particular, para evaluar una estimación se han introducido varios indicadores de precisión, tanto relativos como absolutos, a lo largo de la literatura de estimación de costos. Por ejemplo, error cuadrático medio (MSE), residuos absolutos (AR) o error residual equilibrado (BRE). La revisión de la literatura indicó que los más utilizados, con diferencia significativa, son la "Magnitud media del error relativo de " o MMRE, y la "desviación del error relativo porcentual dentro de x" o PRED(x). De estos dos, el MMRE es el más utilizado. Ambos se basan en el mismo valor básico sin unidades del error relativo de magnitud (MRE) [43].

1.3.2.1 MRE - Magnitude Relative Error

La magnitud del error relativo, para medir el nivel de exactitud, en este caso de las estimaciones de esfuerzos (una a una), se define como:

$$MRE = \left| \hat{E} - E \right| / \hat{E} \qquad (1.\ 2)$$

Donde,

Ê= esfuerzo real
E = esfuerzo estimado

Ahora, teniendo esta primera se puede obtener la media (MMRE) de esta forma:

$$\mathrm{MMRE}(\%) = \frac{1}{n}\sum_{i=1}^{n}\mathrm{MRE} * 100 \qquad (1.\ 3)$$

i = elemento estimado
n = cantidad total de elementos estimados

Se argumenta que MRE es útil porque no penaliza demasiado los proyectos grandes y no tiene unidades (es decir, es independiente de la escala). MMRE se define como el promedio muestral de los MRE. Se consideran una MMRE ≤ .25 como un nivel aceptable de desempeño para modelos de predicción de esfuerzo [43].

1.3.2.2 PRED (r) – Percentage Relative Error Deviation

El porcentaje de desviación del error relativo se define como el porcentaje de elementos estimados cuyo valor relativo de error es menor que r.

$$\mathrm{PRED}(r) = \frac{k}{n} \qquad (1.\ 4)$$

Donde,

n = cantidad total de elementos estimados
k = número de elementos estimados para los cuales se cumple que MRE >= r.

En general, PRED(r) ≥ .75 se considera una precisión aceptable. Contrario a lo que ocurre con MMRE, son deseables valores altos de PRED [42].

2. Capítulo 2: Diseño de Investigación

Este trabajo de investigación se desarrolla dentro del grupo ColSWE (Colectivo de investigación en Ingeniería de Software) de la Universidad Nacional de Colombia con sede en Bogotá, y consiste en la realización de un estudio exploratorio, de tipo no experimental haciendo uso de técnicas cuantitativas.

De esta forma se conoce el estado de la práctica en Colombia, con relación a la estimación de esfuerzo dentro de los proyectos de Desarrollo de software implementados con metodologías ágiles. Para cumplir con este objetivo general, se plantearon los siguientes objetivos específicos:

- ✓ Identificar cuáles son los métodos más utilizados en Colombia para estimar esfuerzo en proyectos ASD y su nivel de precisión.
- ✓ Identificar cuál es el perfil de las personas que estiman esfuerzo en ASD en Colombia.
- ✓ Identificar qué factores pueden afectar la precisión de la estimación de esfuerzo en Colombia para proyectos ASD.
- ✓ Conocer si en Colombia se estima esfuerzo en proyectos ASD de forma similar o diferente a como se hace en otros países, de acuerdo con los resultados obtenidos en este estudio y los trabajos relacionados identificados antes de dar inicio al mismo.

Para el caso colombiano se realizó una encuesta a profesionales con experiencia en estimación de esfuerzo en proyectos de desarrollo ágil de software, con el fin de conocer de qué forma estas metodologías referenciadas en los estudios consultados son aplicadas también en Colombia.

Enseguida se presenta la metodología adoptada para el estudio, así como las preguntas de investigación propuestas y las herramientas utilizadas para dar respuesta a dichas preguntas.

2.1 Estrategia para el desarrollo del estudio

Uno de los objetivos principales de este trabajo es encontrar semejanzas y diferencias entre la forma en que los colombianos estiman esfuerzo en proyectos de desarrollo de software con metodologías ágiles y cómo lo hacen en otros países. Dentro de la revisión de literatura de este estudio, se encontró que en 2015 Usman et al. [17] realizó un estudio de encuesta destinado a validar los resultados de una Revisión Sistemática de Literatura (SLR, Systematic Literature Review,) que fue dirigido a profesionales practicantes de ASD. Por lo tanto, para este estudio se decidió utilizar un enfoque similar al de Usman et al. [17]. Se encontró también una SLR realizada por Fernández-Diego et al [6] que fue publicada seis meses antes del inicio de esta investigación. Aparte de ser una SLR reciente, hay varias circunstancias que la hacen adecuada para el diseño de la encuesta de este estudio. En primer lugar, es una actualización de una SLR realizada por Usman et al. y publicada en 2014 [44]. En segundo lugar, abarcó estudios primarios publicados entre diciembre de 2013 y enero de 2020. Por último, mantiene las mismas cuatro preguntas de investigación de la SLR de 2014 y tres de ellas fueron incluidas en este estudio. Se incluyeron dos preguntas adicionales a la lista para abordar nuestros propios objetivos de investigación.

2.2 Preguntas de investigación

La principal motivación de este estudio es conocer cómo los profesionales colombianos están estimando el esfuerzo en proyectos de tipo ASD. En la ausencia de estudios al respecto al momento de plantear esta investigación, se consideró pertinente y útil realizar un estudio de encuesta que, en el contexto de métodos ágiles, indagara sobre los mismos aspectos investigados por otros estudios a nivel mundial.

Con el enfoque anterior, se adoptó tres de las cuatro preguntas de investigación de Usman [17] y Fernández-Diego [6]. Se incluyó una pregunta relacionada con la precisión de las estimaciones y otra sobre cómo los resultados de este estudio son comparables con estudios similares en otras regiones del mundo.

Como resultado, a través de este estudio, se responden las siguientes preguntas de investigación:

- RQ1: ¿Qué métodos se han utilizado para estimar el esfuerzo en ASD?

- RQ2: ¿Cuál es el nivel de precisión de los métodos de estimación del esfuerzo en ASD?
- RQ3: ¿Qué predictores de esfuerzo se han utilizado para la estimación del esfuerzo en ASD?
- RQ4: ¿Cuáles son las características del conjunto de datos utilizado para la estimación del tamaño o esfuerzo en ASD?
- RQ.5: ¿Existen diferencias significativas entre los resultados de este estudio y estudios previos realizados en otros países?

2.3 La encuesta y los encuestados

Para la selección de la muestra, se planteó un esquema de bola de nieve con profesionales del sector de construcción de software en Colombia.

Para determinar la población objetivo, se hizo una lista de personas que tenían relación académica o laboral con personas del grupo de investigación ColSWE. De igual forma con este grupo inicial se extendió la muestra a personas referenciadas por esta muestra inicial. Además, conociendo de antemano que todos trabajaran para empresas de desarrollo de software en Colombia y que tuvieran experiencia en estimación de esfuerzo en ASD. Este enfoque para seleccionar a los participantes puede clasificarse como “muestreo por conveniencia” o “muestreo por oportunidad”, donde la muestra se toma de un grupo de personas fáciles de contactar.

Lo anterior dio como resultado poder enviar la encuesta en línea a 146 profesionales en desarrollo, analistas de negocios de diversas áreas tecnológicas y regiones de país que han trabajado en proyectos ASD realizando actividades de estimación de software.

El cuestionario contiene 24 preguntas y se divide en seis secciones, teniendo en cuenta los datos demográficos que se requiere conocer y en función de las preguntas de investigación de la siguiente forma:

Una primera sección de preguntas en la que se pretende conocer la edad de las personas que responden la encuesta, el sector para el cual trabajan, el rol que desempeñan en su organización, nivel académico, y tiempo de experiencia que tienen estas personas realizando estimación de esfuerzo en proyectos de desarrollo de software.

En la segunda sección, y con el fin de dar respuesta a la primera pregunta de investigación, se formulan preguntas relacionadas con los métodos que han utilizado estas personas para estimar esfuerzo en proyectos ASD de acuerdo con su experiencia. De igual forma se indaga sobre las actividades asociadas a dichas estimaciones y las herramientas utilizadas.

En la tercera sección del cuestionario se formulan preguntas relacionadas con las técnicas de medición de la exactitud en las estimaciones y cuáles son los factores que afectan frecuentemente dicha exactitud.

En una cuarta sección, y teniendo en cuenta que Planning Poker es la técnica más utilizada para estimar esfuerzo en proyectos ASD [17], se dedica una sección completa para indagar sobre cómo funciona esta técnica para las personas que la usan en función de aspectos como duración, costo y recursos.

En la quinta sección, y con el fin de responder la tercera pregunta de investigación, se formulan preguntas relacionadas con los predictores de esfuerzo utilizados por los encuestados y los aspectos que ellos consideran fundamentales a la hora de estimar proyectos ASD.

Por último, en una sexta sección se plantean preguntas que permiten identificar si los encuestados hacen uso de conjuntos de datos para apoyar sus estimaciones y de ser así, cuáles son y de qué forma los utilizan.

En todas las secciones se proporcionan múltiples opciones de respuesta, teniendo en cuenta que posiblemente estos profesionales no siempre estiman de la misma forma o tienen siempre disponibles las mismas herramientas.

Antes de enviar la encuesta a los 146 profesionales, se realizó un piloto con el diseño inicial del cuestionario, con el fin de obtener realimentación respecto a la claridad de las preguntas y tiempo de respuesta. Dicho piloto se realizó con profesionales estudiantes de posgrado y con experiencia en estimación de proyectos ASD.

La encuesta estuvo disponible durante tres semanas y durante ese período se obtuvieron 60 respuestas completas. Estas respuestas corresponden al 41% de la muestra

inicialmente prevista. La encuesta fue distribuida en idioma español usando Google Forms. Se anexa a este documento versión en español del cuestionario.

2.4 Trabajos relacionados

La revisión de la literatura realizada dentro de este trabajo indica que existen pocos estudios similares a este, y al momento de iniciar esta investigación ninguna publicada aún en Colombia. En esta sección se describen dichos estudios.

El primero de ellos es la investigación de **Usman et al**., que valida un estudio previo de los mismos autores [17]. En este trabajo de 2015, se utilizó una encuesta en línea para recopilar información de profesionales ágiles con experiencia en la estimación del esfuerzo.

Se recopilaron datos de 60 practicantes ágiles de 16 países diferentes pertenecientes a los cinco continentes. En América del sur, los practicantes en ASD eran de Brasil(15) y Argentina(2). Los resultados mostraron que Planning Poker, Analogía y el Juicio de expertos son técnicas de estimación de uso frecuente en ASD, mientras que Story Points-SP es la unidad de medida más empleada. Además, el nivel de conocimientos del equipo y la experiencia previa son los factores de costo más considerados. Más de la mitad de los encuestados cree que sus estimaciones de esfuerzo en promedio están subestimadas/sobreestimadas por un margen del 25 % o más.

La mayoría de los encuestados mencionaron los problemas relacionados con los requerimientos como una de las causas principales de la inexactitud en las estimaciones de esfuerzo en ASD. Las Historias de usuario incompletas y no tener en cuenta adecuadamente los requerimientos no funcionales pueden llevar a subestimar el esfuerzo total. Estos consumen una cantidad considerable de esfuerzo en el diseño y la implementación, por tanto, es importante incorporar su impacto en estimaciones de tamaño y esfuerzo; de lo contrario, es posible que la productividad del equipo pueda resultar mucho más baja de lo esperado.

En cuanto al posible impacto de la elección de la unidad de medida en el margen de error de las estimaciones, el 76% de los encuestados en dicho estudio, reportó una alta precisión (rango de error de 0 a 25 %) seleccionando los Story Points (SP) como métrica de tamaño. Se puede concluir que los SP son una opción preferida para la mayoría de los encuestados que tienen alta precisión en sus estimaciones.

Finalmente, se encontró que la mayoría de los equipos ágiles tienen en cuenta las actividades de implementación y prueba durante la estimación del esfuerzo, y la estimación se realiza principalmente en los niveles de iteración y planificación del *release* en ASD.

El segundo estudio encontrado fue realizado en 2018 por **Vera et al**. [45]. El artículo presenta un estudio empírico que identificó las aproximaciones más frecuentes a la estimación del esfuerzo en las empresas chilenas. Para esto

Definieron criterios de inclusión y exclusión para determinar qué empresas podrían ser invitadas a participar en el estudio. Los criterios de inclusión consideraron que las organizaciones deben ser formalmente una empresa chilena con domicilio en la ciudad de Santiago, tener al menos tres años de antigüedad, tener un negocio estable, operación y desarrollo de software para terceros. Además, tener entre 10 y 49 empleados, con al menos el 75% del personal participando en desarrollo de software, y realizar estimaciones de esfuerzo en sus proyectos de software.

Los principales hallazgos revelan que las empresas utilizan el Juicio de Expertos principalmente debido a su incapacidad para contar con datos históricos confiables. Esta incapacidad para registrar información y datos confiables suele ser una consecuencia del intercambio o rotación de recursos humanos no planificados que ocurre en las pequeñas compañías de software. Por otro lado, los resultados muestran que la mayoría de estas empresas trabajan en proyectos de 3 a 4 meses de duración, y se desarrollan utilizando equipos equivalentes a 3 personas a tiempo completo. Cuando el esfuerzo de desarrollo requerido en el proyecto es más grande, las empresas pretenden dividirlo en subproyectos que se adhieren al tamaño con el que se sienten cómodos. Respecto al método de desarrollo, el 40% de las empresas utilizan un método de desarrollo estructurado, el 40% usa metodologías ágiles y el resto usa un proceso ad hoc. De acuerdo con el registro de entrevistas, en desarrollos estructurados solo hay una persona que estima, mientras que en los desarrollos ágiles muchas de las personas estiman a través de estrategias colaborativas.

Vale la pena señalar que este estudio se centró no solo en los proyectos ASD; también consideró todo tipo de proyectos de desarrollo de software.

En 2019, **Prokopova et al**. [46] realizó un estudio con el objetivo de analizar el proceso de estimación de esfuerzo en proyectos de empresas checas. Un total de 91 empresas,

pertenecientes a seis áreas de tecnologías de la información, fueron encuestadas mediante un cuestionario con doce preguntas. El 39% de las empresas encuestadas pertenecen al sector de desarrollo web y otro 30% al de desarrollo de aplicaciones de escritorio. Por otro lado, El 75% del total de las empresas encuestadas son consideradas pequeñas al tener entre uno y diez trabajadores.
Los hallazgos más relevantes son que el método de estimación más utilizado es el Juicio de Expertos con 57% de incidencia. En general, el 48% de las empresas reportan el tiempo de implementación como un factor clave para la estimación. La causa común (39%) de los retrasos en la entrega de proyectos de software es la volatilidad de los requisitos. En general, el 54% de las empresas no pudo completar una cuarta parte de los proyectos en el plazo estimado. Es importante señalar que este estudio consideró proyectos ágiles y tradicionales.

En cuanto al proceso de estimación de esfuerzo en ASD, **Ramessur et al**. [47] analizó 15 proyectos en una conocida compañía Mauritana y descubrió dieciocho factores que influyen en la estimación del esfuerzo en proyectos ASD. Los factores son los siguientes: partes interesadas, dinámica y compromiso del equipo, comunicación, experiencia del personal y capacidad técnica, experiencia en proyectos anteriores, responsabilidades fuera del proyecto, configuración, seguridad, defectos y cambios durante la implementación, calidad de los requisitos, complejidad de los requisitos, volatilidad de los requisitos, transacciones de datos, facilidad de operación, cambios en los procesos, gestión del proyecto, factores ambientales y tiempo de dedicación. Su hallazgo principal fue que la experiencia del personal y la capacidad técnica, al igual que la complejidad de los requisitos son los dos factores más impactantes en todos los proyectos. Respecto a la experiencia del personal y la capacidad técnica, Ramessur et al. [47] consideran que la estimación del esfuerzo se ve muy afectada por la rotación de personal calificado, también que los equipos con buena experiencia técnica trabajan más eficientemente ahorrando tiempo en la resolución de problemas. Respecto a la complejidad de los requisitos, un requerimiento puede tener una complejidad de nivel bajo, medio o alto para su desarrollo. Dependiendo de su naturaleza, todos los requisitos tienen entonces un nivel de complejidad de desarrollo y así mismo afectan todos los sprints del proyecto ASD.

Por último, durante la realización de este estudio, se conoce de una publicación reciente [48] de la revista ITECKNE de la Universidad Santo Tomás de Bucaramanga, en la que se

muestra los resultados de un estudio realizado con el fin de conocer **el estado de la práctica en la ciudad de Bogotá (Colombia)** respecto a la estimación de esfuerzo en desarrollo de software ágil. Para esto se realizó una encuesta a empresarios de distintas compañías que trabajan en el sector de tecnología informática o estaban en ese momento relacionados con el desarrollo de software ágil, al igual aquellos que han gestionado proyectos ASD. Se encuestó a 413 participantes en la cuidad de Bogotá. La de estructuro en dos grandes secciones: La primera parte recolectó todo el registro de los datos e información básica de los encuestados y, la segunda parte toda la experiencia práctica y el conocimiento de los encuestados en el tema de desarrollo de software mediante metodologías agiles. El autor indica que, debido al tipo de estudio realizado y la muestra, los resultados son generalizables, únicamente para la industria del software de la ciudad de Bogota.

Los principales hallazgos fueron: El 34% de los encuestados consideran de gran importancia la adopción de una metodología de estimación de esfuerzo. El 43,31% considera que los puntos de caso de uso son la técnica de estimación de esfuerzo más utilizada, esto considerando que en la encuesta posiblemente se les planteó que los Use Case Points son una técnica de estimación y no una de medición de tamaño. Por otro lado, para los encuestados el predictor de esfuerzo más utilizado son las líneas de código (38,85%), seguido de los Story Points con un 33,76%. Para esto el autor consideró las métricas de tamaño como predictores de esfuerzo, de acuerdo con lo sugerido por Usman [44]. Como la metodología ágil más adoptada por las empresas se reporta SCRUM, con un 43,95%. Los encuestados consideran la experiencia del equipo como el factor más determinante a la hora de estimar esfuerzo (53,50%).

Respecto a la exactitud de sus estimaciones, se proporcionó a los encuestados una lista de intervalos entre los cuales seleccionaron el porcentaje de error de estimación de esfuerzo considerando el esfuerzo real que se puede evidenciar en el desarrollo de proyectos de software dentro su experiencia. El resultado es que más del 30% de los encuestados consideran que entre el 25 y 50% de sus estimaciones son subestimadas.

Por último, a través de una pregunta abierta se indagó sobre el principal obstáculo que se les presenta a la hora de estimar esfuerzo y se encuentra que son los recursos (26,11%) y el tiempo (23,57%) disponibles.

Finalmente, existen varios estudios sobre estimación de esfuerzo en Historias de Usuario (User Story – US) [49] [50] [51] [52]. Majchrzak et al. [49] propone factores que influyen en la estimación de las historias de usuario. Sus resultados señalan tres grandes grupos de factores: gestión, conocimiento y requisitos. Además, hay factores de defectos posteriores al lanzamiento que están directamente relacionados con estimaciones inexactas: desmotivación, estrés, falta de requisitos y pruebas débiles.

Abrahamsson et al. [50] proponen un método ASD para predecir el esfuerzo de desarrollo basado en historias de usuarios bien estructuradas. Mahniˇc y Hovelja [51] realizaron un estudio empírico utilizando Planning Poker para estimar historias de usuario, cuyo principal hallazgo es la correlación entre la experiencia de los estimadores y la precisión de la estimación. Finalmente, Haugen [52] muestra que la introducción de Planning Poker mejora la estimación del rendimiento del equipo en entornos de programación extrema (Extreme Programming - XP).

3. Capítulo 3: Estado de la práctica en Colombia

A continuación, se relacionan los hallazgos del estudio con base en la encuesta realizada a los profesionales ASD que participaron. En una primera sección se reportan los datos demográficos de los participantes, como formación profesional, roles que desempeñan en las empresas en las que trabajan y su experiencia relacionada con estimación de esfuerzo en proyectos de desarrollo de software. En una segunda sección se reportan los resultados obtenidos con relación a las técnicas de estimación y marcos de metodologías agiles utilizados por los encuestados. En una tercera y cuarta sección, se presentan los resultados obtenidos con relación al nivel de precisión y los factores que afectan las estimaciones realizadas por los encuestados. En una quinta sección se reporta información relacionada con el uso de *datasets* como herramienta de apoyo en el proceso de estimación de esfuerzo. Por último, en una sexta sección se comparan los resultados obtenidos en este estudio, con los resultados reportados en los trabajos relacionados descritos en el capítulo anterior.

3.1 Demografía y perfil del grupo de encuestados

Sesenta (60) profesionales colombianos de diferentes regiones respondieron la encuesta diseñada en este estudio. Geográficamente, la mayor parte de los encuestados (90%) reportaron trabajar en la región andina del país (**Figura 3-1**), donde se encuentran las ciudades más densamente pobladas. Esto concuerda con la información presentada por MinTIC [53], en donde de acuerdo con el Censo del Directorio de Empresas Activas de la Industria del Software y Servicios Asociados con TI de Colombia, para el año 2014 había 4016 empresas activas, 717 de ellas dedicadas al desarrollo de software con un foco mayor en Bogotá y Antioquia como se muestra en la **Tabla 3-1**, que para efectos de este estudio corresponden a la región Andina.

Con relación a la edad de los participantes, el 50% de los encuestados tiene más de 36 años, el 45% tiene entre 25 y 35 años y sólo el 5% de ellos tiene menos de 25 años (**Figura 3-2**).

Figura 3-1: Población demográfica de participantes

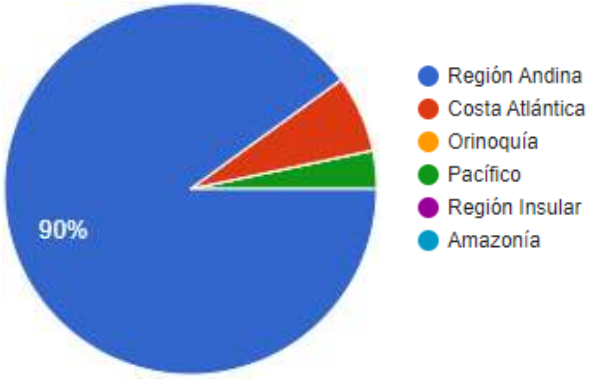


Figura 3-2: Edad de los participantes

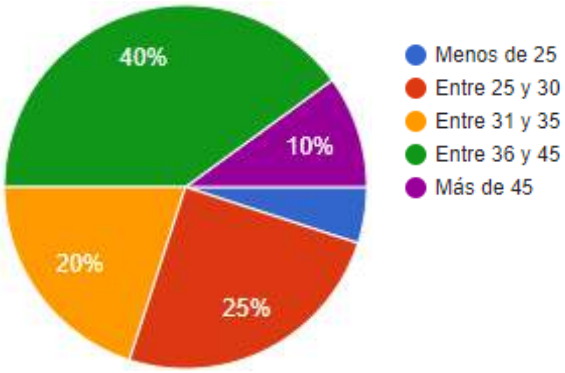


Tabla 3-1: Productos y servicios ofrecidos por número de empresas y departamentos

Ciudad/ Departamento	Producto/servicio						
	Desarrollo / fábrica de software	Manejo de centros de datos (*data center*)	*Testing* de software	Infraestructura como servicio	Consultoría e implementación	Mesas de ayuda (Otras)	Otros
Bogotá	450	528	192	181	105	294	333
Antioquia	119	129	63	43	18	91	66
Atlántico	18	30	14	11	0	9	13
Bolívar	12	6	2	4	0	5	1
Caldas	13	16	8	9	1	7	3
Quindío	7	2	2	0	2	2	6
Risaralda	32	10	2	5	0	4	13
Santander	21	33	6	8	2	14	17
Valle del Cauca	51	49	25	21	8	21	34

Fuente: Estudio MinTIC **[53]**

Realizando la equivalencia entre los hallazgos reportados por MinTIC a nivel de cuidades y departamentos, y los reportados en este estudio por regiones (**Tabla 3-2**), el estudio MinTIC señala entonces que para esa fecha el 89,5% de las 717 empresas clasificadas como Desarrollo/fábrica de software para el año 2014 pertenecían a la región Andina, y es en esta región en donde trabaja el 90% de los participantes en este estudio.

Tabla 3-2: Empresas de Desarrollo por región para el año 2014.

Ciudad / Departamento estudio MinTIC	Cantidad de empresas Desarrollo (MinTIC - año 2014)	Región (Este estudio)
Bogotá	450	Andina
Antioquia	119	Andina
Valle de Cauca	51	Pacífico
Risaralda	32	Andina
Santander	21	Andina
Atlántico	18	Costa Atlántica
Bolívar	12	Costa Atlántica
Caldas	13	Andina
Quindío	7	Andina

3.1.1 Formación y roles

Respecto a la formación profesional de los participantes y los roles desempeñados durante su carrera laboral, más del 90% de ellos posee al menos un pregrado y el 60,3% obtuvo un posgrado (**Figura 3-3**).

Como se muestra en la **Figura 3-4**, gran parte de los encuestados tienen experiencia como desarrolladores de software (46,7 %), diseñadores de software (18,3 %) y analistas de sistemas (11,7%). Hay una representación menor de experiencia otros roles, como arquitectos de software y scrum master (10%), coordinadores de proyectos, gerentes de TI, entre otros.

Figura 3-3: Perfil profesional de los participantes.

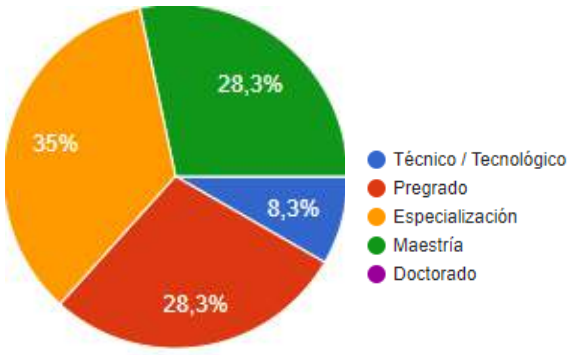
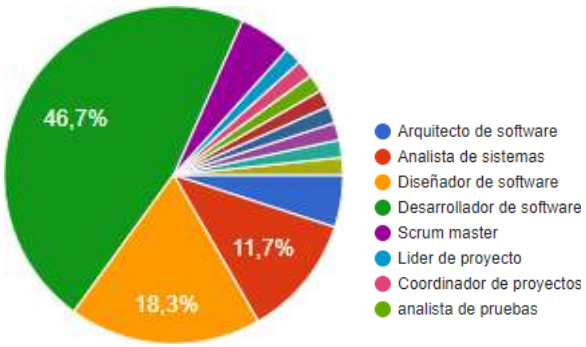


Figura 3-4: Roles de los participantes.



3.1.2 Experiencia profesional y en estimación de esfuerzo

En cuanto a las áreas de negocio en las que se desempeñan los participantes, se encontró que la mayoría lo hace dentro del sector de tecnologías de la información (58,3%) y financiero (26,7%). Otras áreas menores reportadas fueron el sector gubernamental y médico e industrias de la salud.

Un total de 81,7% de los participantes reporta tener más de cinco años de experiencia laborando en la industria de software (**Figura 3-5**). Referente a la experiencia en estimación de esfuerzo, 48,3% tienen más de cinco años de experiencia y 26,7% tienen entre dos y cinco años de experiencia (**Figura 3-6**), estos representan la mayor parte de la muestra.

Figura 3-5: Experiencia en la industria de desarrollo de software

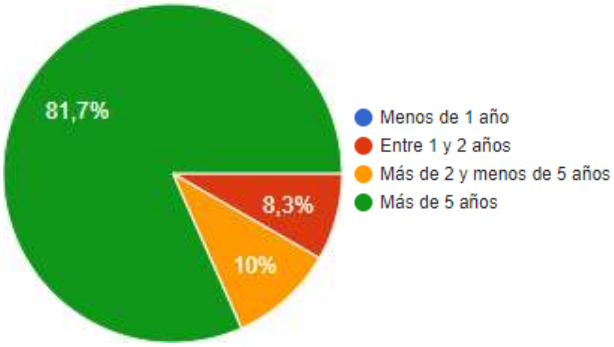
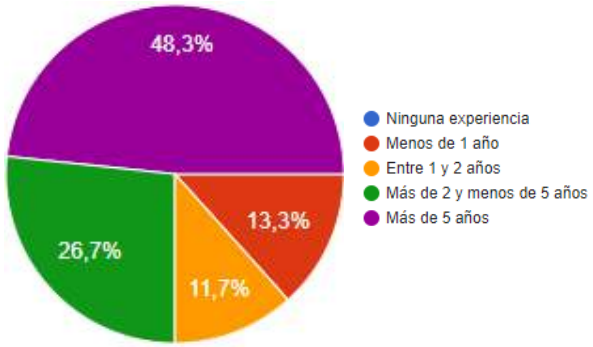


Figura 3-6: Experiencia realizando estimación de esfuerzo



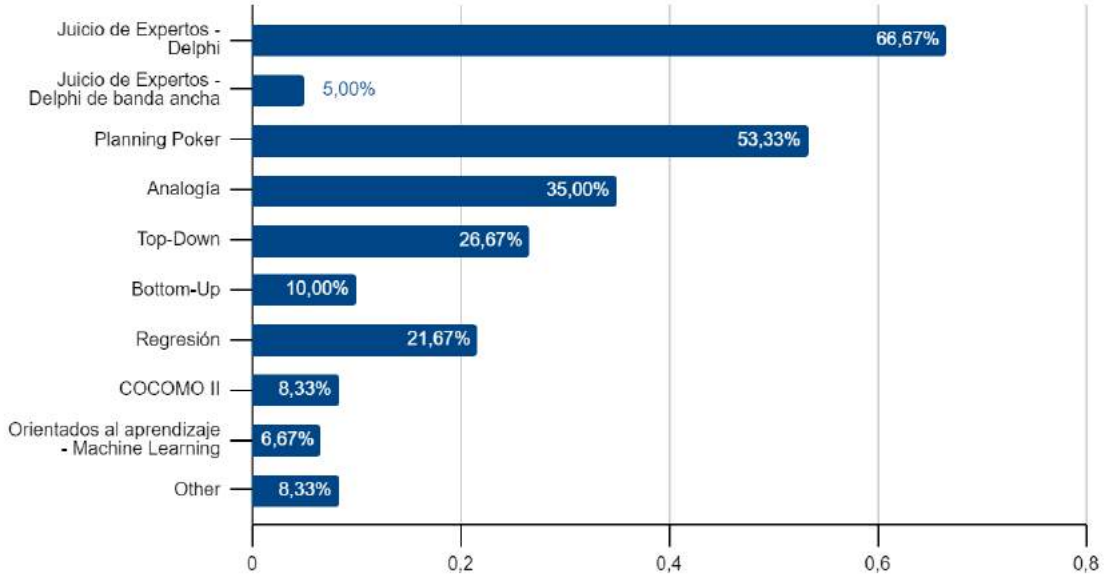
3.2 Estimación de esfuerzo

Respecto a qué métodos de estimación de esfuerzo usan los desarrolladores colombianos dentro de sus proyectos ASD; en la encuesta hay seis preguntas orientadas a este aspecto. Al contestar cada pregunta sobre estimación del esfuerzo, los encuestados pueden elegir una o más respuestas de una lista de opciones sugeridas. Así mismo, el cuestionario en línea les permite dar una o más respuestas adicionales no incluidas en la lista provista dentro del ejercicio de encuesta de este estudio.

3.2.1 Técnicas de estimación de esfuerzo

A los encuestados se les proporcionó una lista de nueve técnicas sugeridas, donde seleccionaron las que han utilizado. Los resultados muestran que las cinco técnicas más utilizadas son Juicio de expertos (66,7 %), Planning Poker (53,3 %), Analogía (35%), Top-Down (26,7%) y Regresión (21,7%). La **Figura 3-7** muestra los resultados para todas las técnicas de estimación esfuerzo sugeridas: Juicio de Expertos – Delphi, Juicio de Expertos - Delphi de banda ancha, Planning Poker, Analogía, Top-Down, Bottom-Up, Regresión, COCOMO II (Constructive Cost Model), Orientados al aprendizaje (Machine learning), Otros. Dentro de esta opción “otros”, un participante manifiesta haber utilizado un método propio de la empresa en la que labora basado en COCOMO I; otro participante considera los Puntos funcionales como una técnica de estimación y otro ha usado el Juicio de expertos, pero no dentro de las técnicas Delphi mencionadas.

Figura 3-7: Técnicas de estimación de esfuerzo

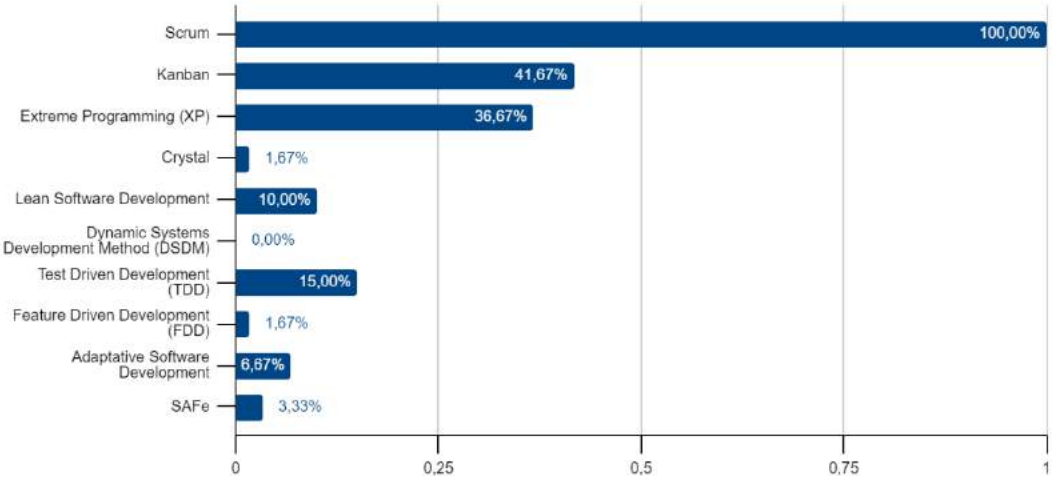


3.2.2 Marcos utilizados para ASD

En segundo lugar, se preguntó sobre los métodos ágiles utilizados por los encuestados. El cuestionario proporciona una lista de diez metodologías ágiles sugeridas, donde los encuestados indicaron las que han experimentado con la posibilidad de escoger más de una. Los métodos más utilizados son Scrum (100%), Kanban (41,7%), Extreme Programming (36,7%) y Test-Driven Development (15%). Menos del 15% de los

encuestados utilizan otros métodos ágiles, incluido el desarrollo de software Lean, Crystal, Desarrollo basado en funciones (FDD), Desarrollo adaptativo de software. Marcos de trabajo para escalamiento ágil como SAFe fue reportado solo en unos pocos casos. (**Figura 3-8**).

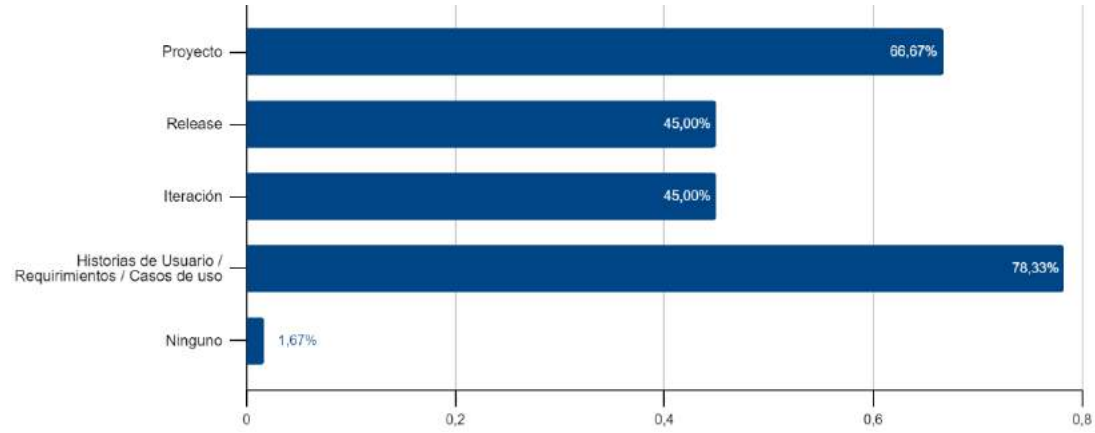
Figura 3-8: Marcos ASD.



3.2.3 Niveles de planificación y estimación por actividades

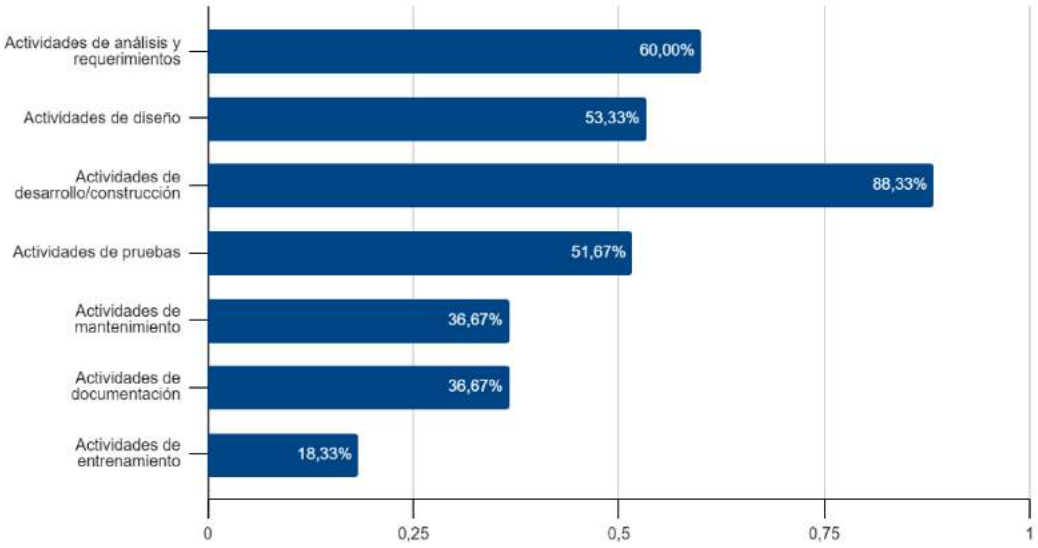
En tercer lugar, se preguntó en qué niveles de planificación realizan los encuestados la estimación del esfuerzo. Ellos eligieron entre cinco niveles de planificación sugeridos con la posibilidad de seleccionar más de un nivel. Los resultados de la **Figura 3-9** indican que el 78,3% de los encuestados estiman a nivel de Historias de Usuario, el 66,7% estima a nivel de proyecto y el 45% estima a nivel de lanzamiento de *release* e iteración.

Figura 3-9: Estimación de esfuerzo por nivel de planificación.



En cuarto lugar, se preguntó sobre las actividades del ciclo de vida del software en para las cuales los encuestados realizaron una estimación del esfuerzo. Cada encuestado pudo elegir entre siete grupos de actividades sugeridos. Las actividades de software más relevantes cuyo esfuerzo puede ser estimado son desarrollo (88,3%), seguido de análisis (60%), diseño (53,3%), pruebas (51,7%), documentación y mantenimiento (36,7%) y capacitación (18,3%). La **Figura 3-10** muestra las actividades de ingeniería de software incluidas en las estimaciones de esfuerzo.

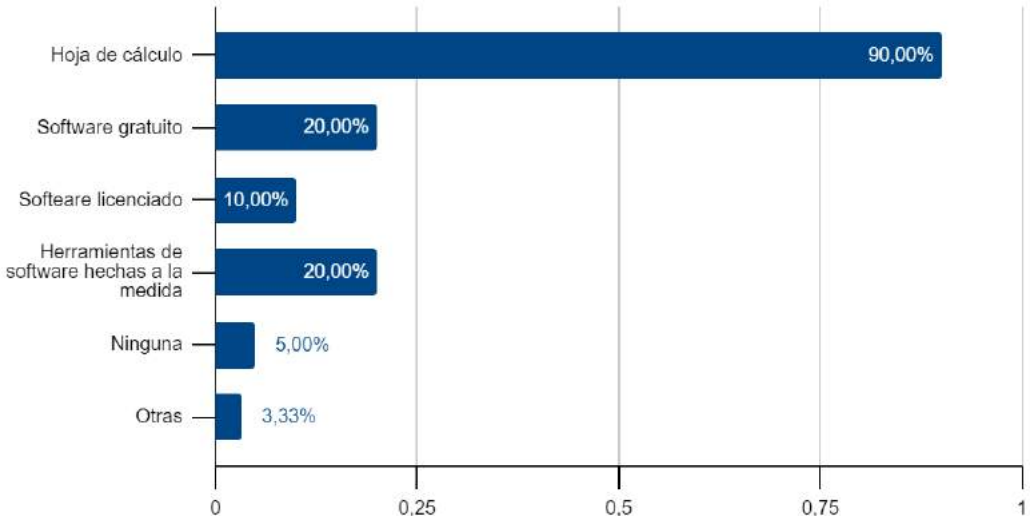
Figura 3-10: Estimación de esfuerzo por actividades ASD.



3.2.4 Herramientas de apoyo en proceso de estimación

Se indagó sobre las herramientas que podrían utilizar los participantes para apoyar el proceso de estimación del esfuerzo. A los encuestados se les proporcionó una lista de siete tipos de herramientas. Se encontró que las hojas de cálculo son las más utilizadas (90%), seguidas de las herramientas de software gratuitas y personalizadas (20%). Los resultados también muestran que las herramientas de software con licencia se utilizan en menor medida (10%). El 5% de los encuestados no utiliza ninguna herramienta y solo el 1,7% utiliza la gestión de proyectos o herramientas de gestión del ciclo de vida de las aplicaciones, como Kanban, MS Project o Azure DevOps. (**Figura 3-11**)

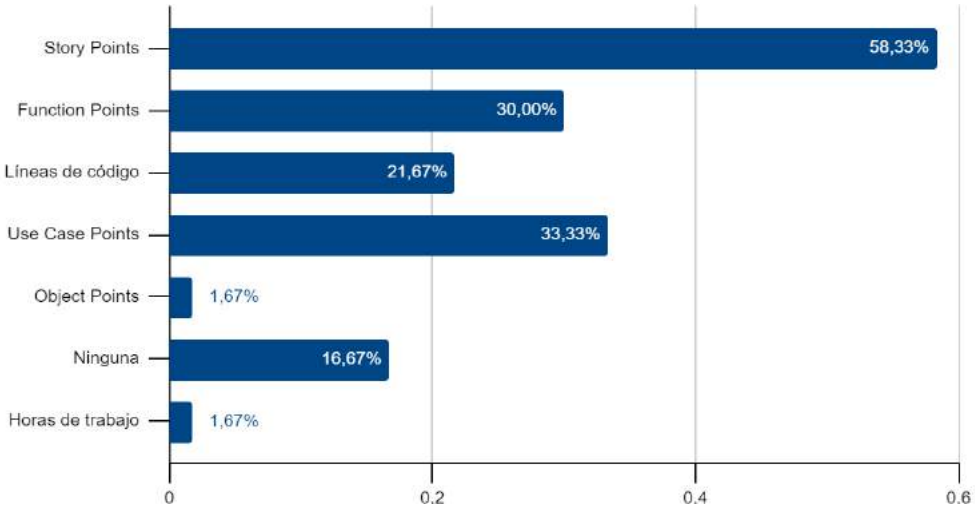
Figura 3-11: Herramientas de apoyo



3.2.5 Técnicas de medición

Finalmente, en esta sección asociada a los métodos utilizados para estimar esfuerzo en ASD, se preguntó sobre las técnicas de medición del software a las que acuden los participantes dentro del proceso de estimación de esfuerzo. Los encuestados eligieron entre siete técnicas sugeridas. La técnica más utilizada es Story Points (58,3%), seguida de Use Case Points (33,3%), Puntos de función (30%) y Líneas de código (21,7%). Un porcentaje significativo de los encuestados (16,7%) no utiliza ninguna técnica y uno de ellos reportó "horas de trabajo" como una técnica de medición adicional (**Figura 3-12**).

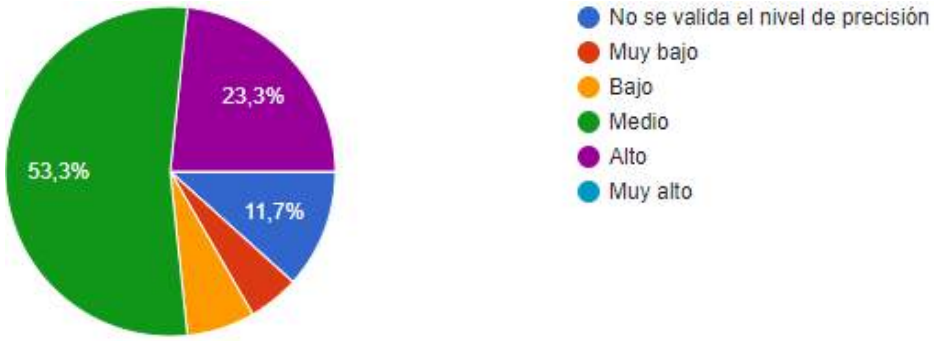
Figura 3-12: Técnicas de medición.



3.3 Nivel de exactitud de las estimaciones

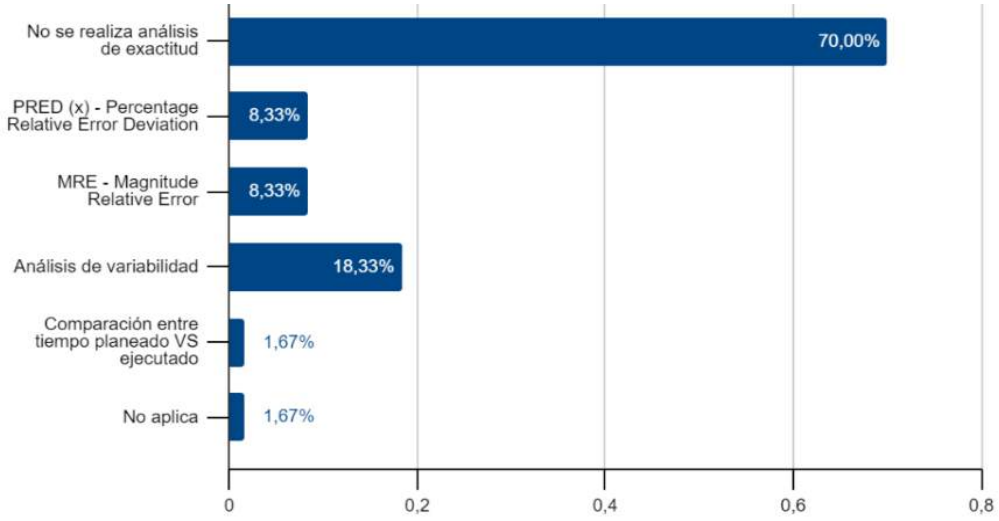
Para responder a la pregunta de investigación relacionada con el nivel de exactitud de las estimaciones realizadas por los participantes, la encuesta incluye tres preguntas generales y una específica para la técnica de Planning Poker. Los resultados muestran que el nivel de precisión es alto para el 23,3% de los encuestados, medio para el 53,3% y bajo o muy bajo para el 11,7% de ellos. Adicionalmente, el 11,7% de los encuestados no evalúa el nivel de precisión de sus estimaciones (**Figura 3-13**).

Figura 3-13: Nivel de exactitud de las estimaciones.



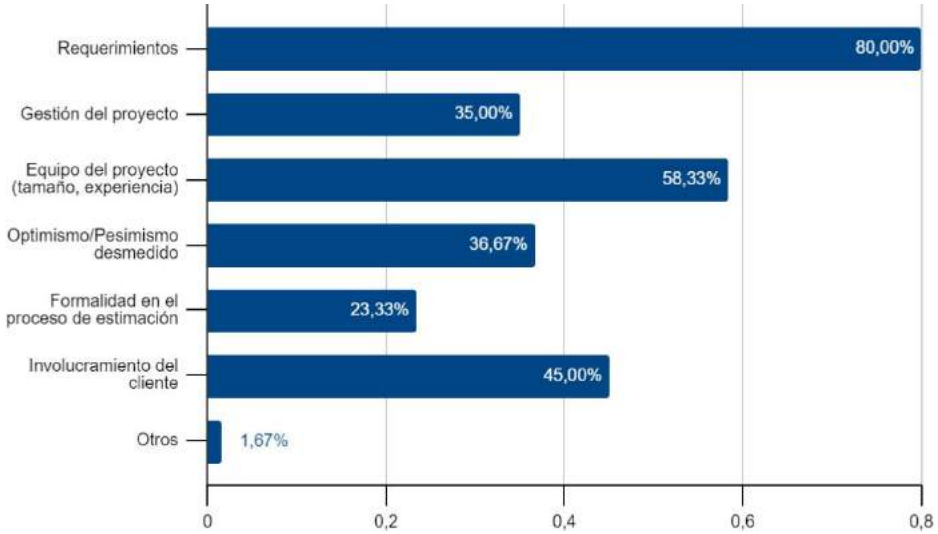
Con respecto a las técnicas para analizar la precisión de la estimación del esfuerzo, cada encuestado eligió entre cuatro técnicas posibles (**Figura 3-14**). El 70% de los encuestados no utiliza ninguna técnica para calcular los niveles de precisión de la estimación, mientras que el 18,3% utilizan análisis de variabilidad, 8,3% utilizan PRED(X) y MRE.

Figura 3-14: Técnicas para analizar nivel de precisión.



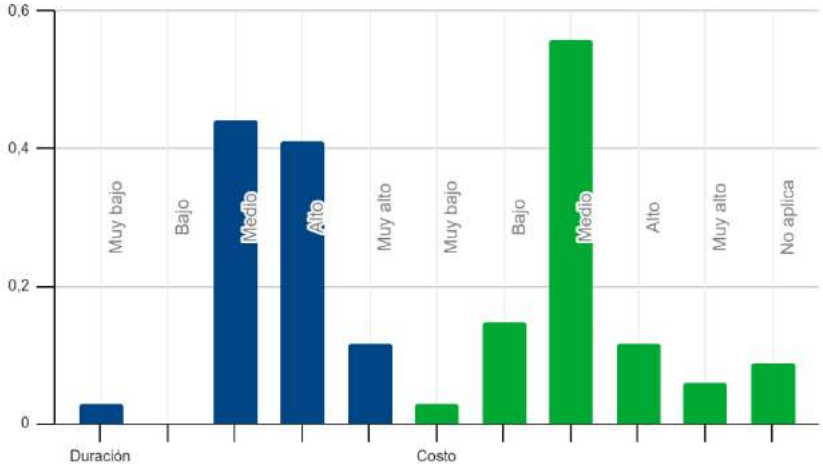
Respecto a los factores que afectan la precisión de la estimación del esfuerzo, cada encuestado eligió entre seis factores sugeridos. Los resultados muestran que los factores más relevantes son la definición de requisitos (80%), seguido del tamaño y la experiencia del equipo del proyecto (58,3 %), involucramiento del cliente (45 %), exceso de optimismo o pesimismo (36,7 %), gestión del proyecto (35%) y formalidad del proceso de estimación (23,3%). La **Figura 3-15** muestra todos los factores que de alguna manera influyen en el nivel de precisión.

Figura 3-15: Factores que influyen en el nivel de exactitud



Teniendo en cuenta que Planning Poker es la segunda técnica de estimación más utilizada (56,7%) en Colombia después del Juicio de Expertos y la más utilizada en el mundo [17], se consideró importante profundizar a cerca del nivel de exactitud cuando se utiliza esta técnica para estimar otros aspectos. Para esto, dentro del grupo de encuestados que informó haber utilizado Planning Poker como técnica de estimación de esfuerzo, se realizó otra pregunta para conocer qué tan efectiva (nivel de exactitud) ha sido su utilización para estimar aspectos del proyecto además del esfuerzo, como lo son la duración y el costo de los proyectos basados en dicha estimación (**Figura 3-16**).

Figura 3-16: Efectividad de Planning Poker para estimar duración y costos.



El 56% de los encuestados consideran que el nivel de exactitud al utilizar Planning Poker como técnica de estimación del costo es medio, seguido de un 44% que consideran también un nivel medio a la hora de estimar la duración haciendo uso de esta técnica. (**Tabla 3-3**).

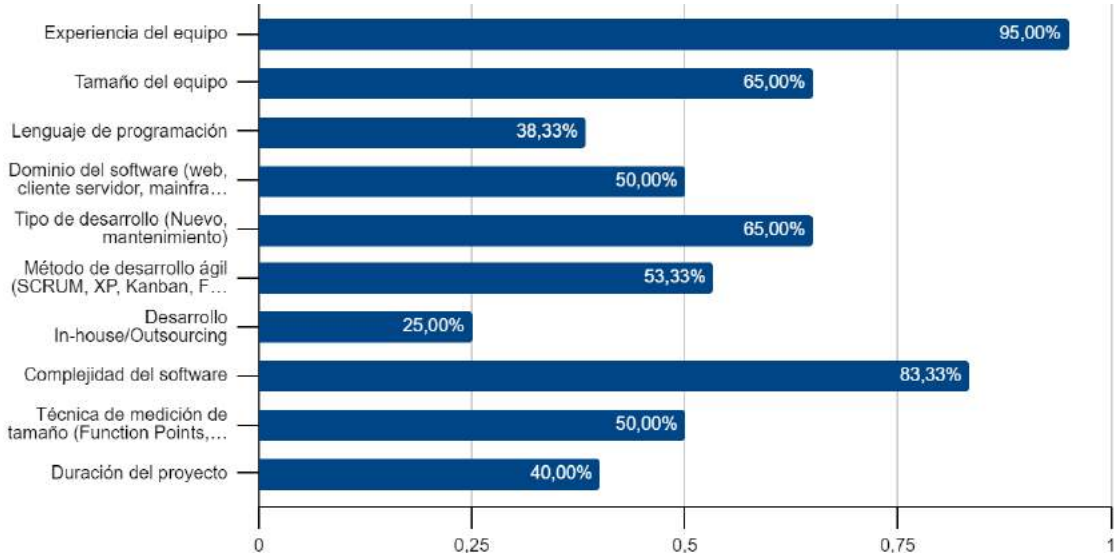
Tabla 3-3: Efectividad de Planning Poker para estimar duración y costos**.**

Tipo de Estimación	Nivel de exactitud					
	Muy alto	Alto	Medio	Bajo	Muy bajo	No aplica
Duración	12%	41%	44%	0%	3%	0%
Costo	6%	12%	56%	14%	3%	9%

3.4 Predictores de esfuerzo

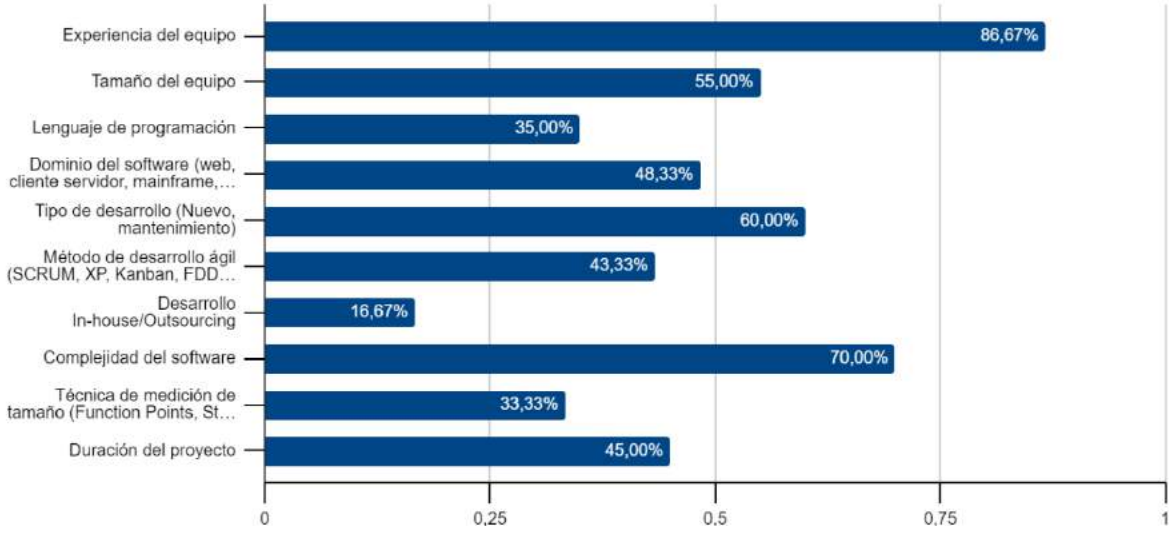
Este estudio tiene también como objetivo investigar los factores que afectan las estimaciones de esfuerzo. Se formularon dos preguntas para saber cuáles son los predictores de esfuerzo considerados fundamentales en los proyectos ASD y cuáles los realmente tenidos en cuenta por los encuestados a la hora de realizar estimaciones dentro de sus proyectos. En cuanto a los predictores de esfuerzo, los encuestados consideran que ciertas variables al estimar esfuerzo son determinantes, tales como experiencia del equipo como principal factor (95%), seguido de la complejidad del software (83,3%), tipo de desarrollo (nuevo o mantenimiento) y tamaño del equipo (65%). Se consideran otros factores, pero en menor medida, incluido el método de desarrollo (ágil o tradicional), el dominio del software y el lenguaje de programación, entre otros. (**Figura 3-17**)

Figura 3-17: Predictores de esfuerzo que afectan la estimación.



En cuanto a los predictores realmente utilizados por los encuestados, la lista resultante es casi la misma. Los predictores de esfuerzo más importantes son los más utilizados en la práctica por los encuestados. Dos diferencias notables se identifican en este escenario, por un lado, el método ágil utilizado, y por otro, la duración del proyecto. El enfoque ágil adoptado por el equipo parece ser menos importante en la práctica, mientras que la duración del proyecto resulta ser más importante en la práctica que en la teoría. (**Figura 3-18**)

Figura 3-18: Predictores utilizados a la hora de estimar esfuerzo.



3.5 Datasets para estimar esfuerzo

En primer lugar, el hallazgo más importante es que la mayoría de las personas encuestadas (85%) nunca ha realizado un cálculo del esfuerzo basado en conjuntos de datos. Sin embargo, a partir de las respuestas de las personas que sí han usado este método para apoyar la estimación de esfuerzo (nueve personas – 15%), se identificaron dos características de los conjuntos de datos (*datasets*) utilizados por ellos.

La primera es que las organizaciones en las que trabajan los encuestados que han manifestado utilizarlos, son en su mayoría dueñas de estos *datasets*. La segunda es que estos son utilizados como apoyo para estimar esfuerzo por analogía. Además, solo dos de los nueve participantes que han utilizado *datasets* han acudido a la adquisición de licencia para esto, cuatro más los han utilizado de libre acceso y por último seis más han utilizado *datasets* propiedad de la empresa (posiblemente construidos con registros de proyectos anteriores).

En cuanto a cómo se han utilizado los *datasets*, se encontró que siete participantes los utilizan para hacer comparaciones entre proyectos de una misma empresa, cinco participantes aplican técnicas de estimación de esfuerzo utilizando *datasets*, y tres utilizan los datasets para estimar el esfuerzo en función de las comparaciones con proyectos fuera de sus empresas.

3.6 Semejanzas y diferencias con otros estudios

Dentro de las motivaciones de este estudio se encuentra qué tan diferente o semejante es la forma en que los desarrolladores colombianos realizan estimación de esfuerzo en proyectos ASD respecto al resto del mundo. De igual forma, se pretende conocer en qué medida acuden a procedimientos o técnicas que ya son internacionalmente utilizadas.

Los trabajos relacionados con este estudio inicialmente son todos realizados por fuera de Colombia [17] [45] [46]. Sin embargo, con posterioridad al inicio de este estudio, se publicó un estudio realizado con el fin de conocer el estado de la práctica en la ciudad de Bogotá (Colombia) respecto a la estimación de esfuerzo en desarrollo de software ágil [48]. Teniendo en cuenta los aspectos tratados en cada uno de dichos trabajos de investigación, incluido este último estudio mencionado, se consideran todos comparables con este

trabajo de investigación. Respecto al trabajo colombiano realizado con una muestra de la ciudad Bogotá [48], en esta sección se realiza inicialmente una comparación con los resultados de este estudio con el fin de verificar sí los resultados presentan la misma tendencia, teniendo en cuenta que en Bogotá se encuentra concentrada la mayor población dedicada al desarrollo de software [53]. Posteriormente se presenta entonces la comparación entre los resultados de todos los trabajos considerados y este estudio.

La **Tabla 3-4** muestra el resumen de los estudios relacionados considerados en esta investigación en cuanto a las características utilizadas para recolectar datos y la muestra considerada.

Tabla 3-4: Características de trabajos relacionados.

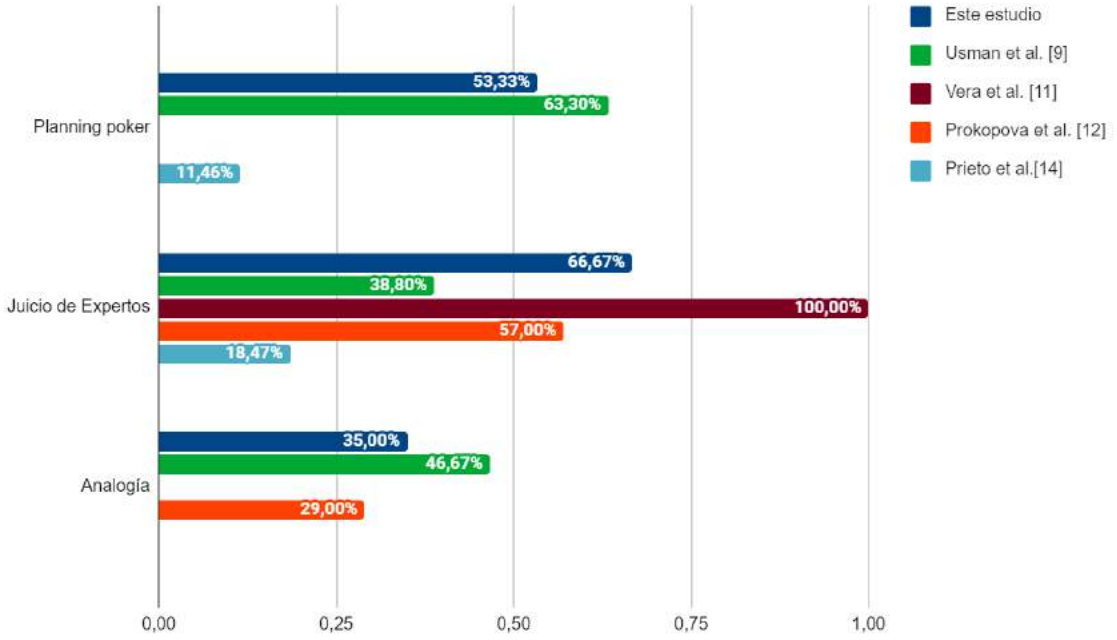
Estudio	Instrumento	Unidad de Análisis	País / Ciudad	Características
Usman et al. [17]	Encuesta	Individual	Brasil, Argentina, USA, Canadá, Pakistán, India, Australia, Francia, Irlanda, Estonia, Dinamarca, Italia, Ucrania, Suecia, Reino unido	Cuestionario online 60 personas encuestadas
Vera et al. [45]	Entrevistas	Compañía	Chile	Entrevista semi estructurada 10 personas entrevistadas
Prokopova et al. [46]	Encuesta	Compañía	República Checa	Cuestionario enviado a 91 compañías
Prieto et al. [48]	Encuesta	Compañía	Colombia / Bogotá	314 empresas distribuidas así: 205 empresas de desarrollo 109 empresas de otros sectores
Este estudio	Encuesta	Individual	Colombia /Región Andina, Atlántica, Pacífica	Cuestionario online 60 personas encuestadas

3.6.1 Técnicas de estimación

Con respecto a las técnicas de estimación de esfuerzo, Usman et al. [17] concluyen que Planning Poker y Analogía son los Métodos más utilizados en ASD. Vera et al. [45] reportan el Juicio de Expertos como el método principal para la estimación del esfuerzo del software y, en algunos casos, Juicio de Expertos combinado con Analogía. En este estudio se concluye que, el Juicio de Expertos y Planning Poker son los métodos más utilizados para la estimación del esfuerzo. Este estudio y el de Usman et al. [17] son totalmente comparables, ya que los encuestados eligieron de la misma lista de métodos de estimación. La comparación con el estudio de Vera et al. [45] es parcial dado que la lista de métodos proporcionada por su encuesta es desconocida. En el estudio de Prokopova et al. [46], los encuestados tenían la opción de elegir Juicio de Expertos y Analogía, pero

no Planning Poker. Respecto al caso Bogotá, sólo son comparables Planning Poker y Juicio de Expertos. Lo anterior quiere decir que sólo los resultados obtenidos en relación con el Juicio de Expertos son comparables en todos los estudios, como se muestra en la **Figura 3-19**.

Figura 3-19: Métodos de estimación en estudios relacionados.



Respecto a las técnicas consideradas, y de acuerdo con lo reportado en los estudios relacionados, el Juicio de expertos es la técnica más utilizada a la hora de realizar una estimación de esfuerzo, excepto para Prieto et al. [48] y esto se explica debido a que en dicho estudio además se ofreció *Use Case Points* y *Story Points* como técnicas de estimación y es en estas dos opciones en donde se agrupa casi el 70% de las respuestas obtenidas [48]. Para determinar si los estudios relacionados son o no comparables con este estudio, se consideró que las técnicas a comparar estuvieran relacionadas en al menos tres de los cinco estudios graficados.

3.6.2 Predictores de esfuerzo

En relación con los predictores de esfuerzo, los resultados de este estudio señalan que el más relevante es la experiencia del equipo, seguido de la complejidad del software y tipo de desarrollo (nuevo o de mantenimiento). Ramessur et al. [47] enumera factores similares que afectan al software estimaciones de esfuerzo y concluye que los factores principales que pueden afectar la precisión de la estimación son la experiencia del equipo y habilidades técnicas, complejidad de los requisitos.

En Usman et al. [17] se consideran los predictores de esfuerzo de dos formas: Pueden ser métricas de tamaño o generadores de costos. Para efectos de comparar dicho estudio con este, los predictores de esfuerzo de este estudio corresponden a generadores de costos en Usman et al. [17].

Para Prokopova et al. [46] se consideraron algunos factores que pueden ser determinantes a la hora de estimar el valor de un proyecto, pero no con relación al esfuerzo requerido. Estos difieren de lo que se consideran predictores de esfuerzo en los demás trabajos relacionados, por tanto, no son comparables con los considerados en este estudio. Dichos factores son: comparación con otros proyectos, tiempo esperado de desarrollo, capacidad, competencia, calidad de los documentos, solvencia del cliente y por último si el desarrollo necesita o no subcontratistas.

La **Tabla 3-5** muestra los factores más relevantes dentro de los resultados obtenidos en este estudio y algunos considerados en los estudios relacionados.

Tabla 3-5: Factores que afectan la estimación de esfuerzo.

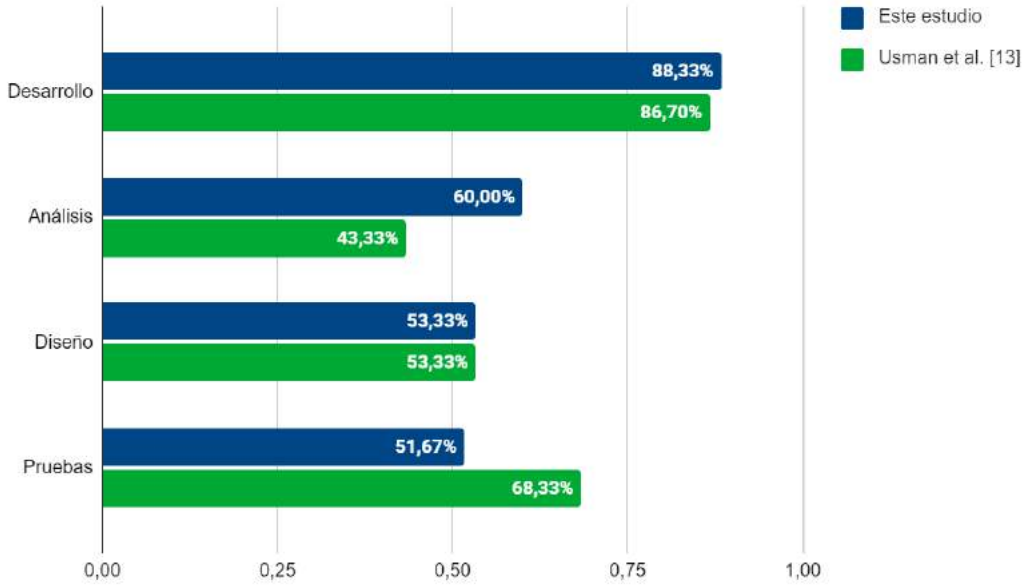
Factor	Este estudio	Usman et al. [17]	Ramessur et al. [47]	Prieto et al. [48]
Experiencia del equipo	95%	71,67%%	93,75%	53,5%
Complejidad del software	83,33%	N/A	93,75%	N/A
Tipo de desarrollo	65%	N/A	N/A	N/A
Tamaño del equipo	65%	N/A	N/A	N/A
Requerimientos no Funcionales	N/A	40%	N/A	24,84%
Tamaño de la tarea	N/A	41,67	N/A	21,66%

Con respecto a los factores que afectan la estimación del esfuerzo, son similares en todos los trabajos relacionados. En consecuencia, la relevancia asignada a estos factores también es similar. Finalmente, la experiencia del equipo es el factor más importante para considerar en la estimación del esfuerzo según los resultados de este estudio y todos los estudios relacionados [17] [47] [48] [49].

3.6.3 Actividades Estimadas

Los resultados de este estudio muestran que casi todos los encuestados (88,33%) estiman actividades de codificación principalmente, mientras que el análisis (60%), diseño (53,33%) y las actividades de pruebas (51,67%) se tienen en cuenta para la estimación del esfuerzo en proporciones menores pero similares. Esto concuerda con Usman et al. [17], donde las actividades de codificación se consideran las más importantes para la estimación del esfuerzo (86,67%). Sin embargo, encontraron que las actividades de análisis (43,33%) y diseño (53,33%) son consideradas en menor medida para estimar esfuerzo, mientras que las actividades de prueba son el segundo factor más importante. La **Figura 3-20** muestra la comparación mencionada anteriormente. En los otros dos estudios relacionados, este hecho no fue considerado, por lo tanto, la comparación no es posible.

Figura 3-20: Actividades estimadas.



3.6.4 Precisión de la estimación

En cuanto a la precisión de la estimación, en este estudio se clasifica su nivel con una escala de cinco posibles valores: muy alto, alto, medio, bajo y muy bajo. Con el mismo propósito, Usman et al. [17] utiliza dos categorías principales: sobreestimado o subestimado para expresar la precisión de la estimación y así mismo lo hace Prieto et al. [48].

Vera et al. [14] no informan cuán precisas son las estimaciones de esfuerzo, sino que concluyeron que el esfuerzo siempre se sobrestima debido a un factor multiplicador que magnifica las estimaciones. En el estudio realizado por Prokopova et al. [20], solo se reporta el porcentaje de proyectos no entregados a tiempo.

Los resultados de este estudio indican que el nivel de precisión de las estimaciones es alto para el 23,3% de los encuestados, medio para el 53,33%, y bajo o muy bajo para el 11,67% de ellos. Por otro lado, el 11,67 % de los encuestados no evalúa el nivel de exactitud de sus estimaciones. Los resultados de Usman et al. [17] sugieren que las estimaciones de esfuerzo de alrededor de la mitad de los equipos ágiles son imprecisas (subestimadas) por un factor de 25% o más. Para esto se dividieron los rangos de error de estimación en dos grandes grupos: rango de error de 0 a 25 % (subestimado y sobreestimado) y más del 50% para la subestimación. El primer grupo puede representar un nivel de precisión aceptable, mientras que el segundo representa una preocupación para los equipos ASD.
Por otro lado, pero analizado de la misma forma, Prieto et al. [48] reporta que para el 47% de los encuestados, la estimación de esfuerzo en el desarrollo de software se encuentra sub/sobreestimado por un factor contemplado entre el 25% y el 50%, en el cual la tendencia dominante es subestimada con 33%.

Vera et al. [45] indican que casi la mitad de las empresas sobrestiman el esfuerzo del proyecto, independientemente del ciclo de vida que se está utilizando. Esto sucede en algunos casos porque la empresa utiliza un multiplicador para sobreestimar los proyectos con el fin de para reducir el riesgo del esfuerzo subestimado. Además, las empresas no obtienen realimentación de sus estimaciones y, en consecuencia, no conoce el nivel de precisión de estas.
Prokopova et al. [46] investigan la cantidad de proyectos que se entregan a tiempo. Encontraron que más de la mitad de las empresas (54%) no terminan el 25% de sus proyectos dentro de sus estimaciones, lo cual es consistente con la tendencia identificada en los otros estudios relacionados.

La **Tabla 3-6** consolida los hallazgos de cada estudio comparados en función de si el nivel de exactitud es alto, medio o bajo. Para esto se realizó una equivalencia entre la escala de valores definida en este estudio para clasificar el nivel de precisión (Alto/Medio/Bajo/Muy bajo/No se valida) y las categorías definidas por Usman et al. [17] y Prieto et al. [48] para

clasificar el margen de error dentro del cual los encuestados consideraron se encontraban sus estimaciones bien por subestimación o sobreestimación.

Tabla 3-6: Nivel de precisión por estudio.

Nivel de precisión	Este estudio	Margen de Error (equivalencia)	Usman et al. [17]	Prieto et al. [48]
No se valida	11,67%		N/A	N/A
Muy bajo	5,00%	50%-100% - subestimado	6,67%	2,55%
Bajo	6,67%	25%-50% - sobreestimado y subestimado	45%	47,7%
Medio	53,33%	5%-25% - sobreestimado y subestimado	26,66%	45,86%
Alto	23,33%	0-5% - difiere de la estimación real	21,67%	3,82%

Los resultados muestran que los tres estudios tienen en común qué: Una parte muy pequeña de los encuestados (5%, 6,67%, y 2,55%) consideran que el nivel de precisión de sus estimaciones es *muy bajo*.

Los resultados de este estudio muestran que sólo el 6,67% de los colombianos encuestados consideran *bajo* el nivel de precisión de sus estimaciones, contrario a lo que se evidencia en [17] y [48] donde una gran parte de los encuestados (45% y 47%) consideran *bajo* el nivel de precisión de sus estimaciones. Por último, se evidencia que la mayoría de los encuestados en este estudio consideran *medio* y *alto* el nivel de precisión de sus estimaciones (76,66%) mientras que en [17] y [48] lo hacen aproximadamente el 50% de los encuestados.

De estos resultados se puede inferir que en Colombia se cuenta con un nivel de precisión más alto a la hora de realizar estimaciones de esfuerzo, respecto a lo reportado por los otros estudios, sin embargo, este resultado se puede deber a dos razones no concluyentes pero válidas. La primera de ellas puede ser la diferencia de escalas establecidas para clasificar la precisión en este estudio y en los estudios relacionados, pues lo que para unos puede ser considerado como una estimación como precisión alta, para otros no lo es tanto si esto se enmarca en un rango de porcentaje de error o acierto a la hora de estimar esfuerzo. La segunda razón puede ser el rol de los encuestados dentro de los proyectos ASD, pues en este estudio la mayor parte de los encuestados son desarrolladores y diseñadores de software (65%) (**Figura 3-21**), quienes tal vez tienden a creer que sus estimaciones son buenas, sin embargo, es probable que otros *stakeholders* del proyecto tengan opiniones menos optimistas con respecto a la precisión. Por otro lado, en Usman

et al. [17] la proporción de desarrolladores y diseñadores de software es sólo del (18,4%) (**Figura 3-22**).

Figura 3-21: Roles de los participantes de este estudio

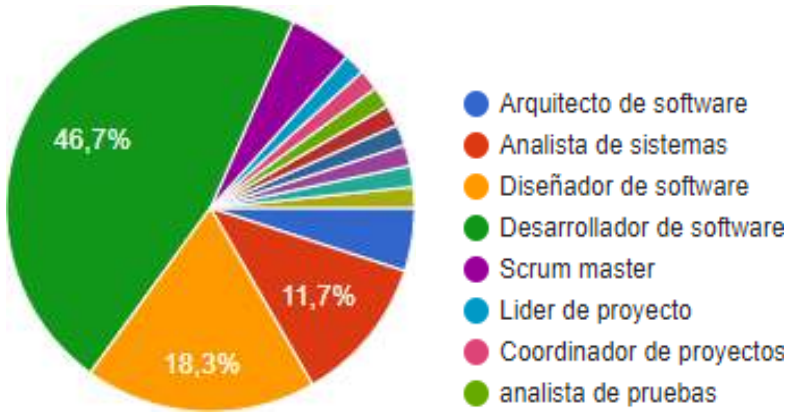
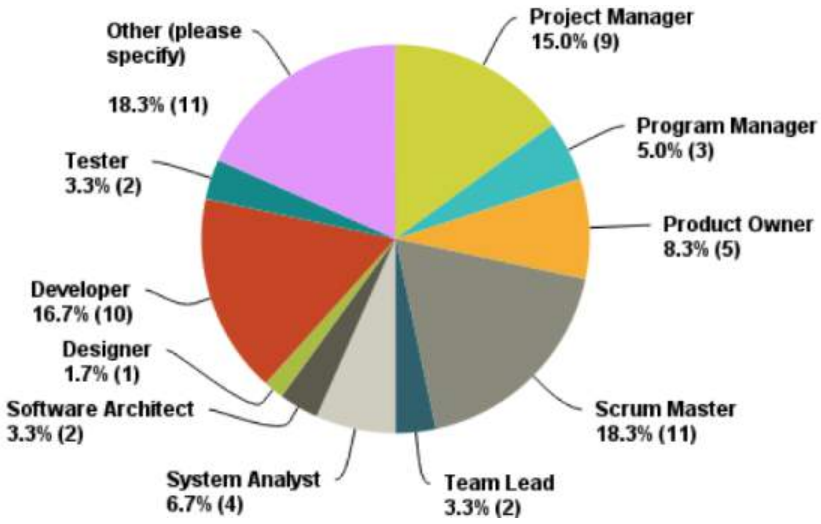


Figura 3-22: Roles de los participantes en Usman et al. [17]



3.7 Amenazas a la validez

En términos generales, el valor de los resultados de un estudio empírico depende de su validez. La validez determina cuánto podemos confiar en que los hallazgos del estudio reflejan con precisión la realidad que queremos describir o explicar. Para el caso particular de una encuesta, muchos factores pueden afectar esa validez y se discuten en esta sección, así como, la acciones que se tomaron para evitar o mitigar sus efectos.

La validez externa es la medida en que los resultados de este estudio pueden generalizarse a otros individuos, entornos, y tiempos.

En primer lugar, los participantes eran en su mayoría desarrolladores o personas que desempeñan roles específicos dentro de los equipos de software como diseñadores, analistas, arquitectos o scrum masters. La mayoría de ellos tienen más de 5 años de experiencia en la industria del software, con edades que van desde menos de 25 hasta más de 45 años. Aunque la muestra fue a conveniencia del estudio, es bastante heterogénea en términos de experiencia, edad y rol de trabajo. En segundo lugar, la gran mayoría de ellos están en la región andina de Colombia donde se encuentra concentrada la mayor población y el mayor desarrollo industrial del país. Por lo tanto, se considera que este estudio tiene una amplia validez externa en función de la población objeto de estudio, para que los resultados se extiendan a la población de desarrolladores en Colombia.

Las amenazas a la validez interna se refieren a factores extraños que pueden introducir errores o sesgos que finalmente conducen a resultados que distan mucho de la realidad. En primer lugar, para reducir la amenaza de que los sujetos no sean lo suficientemente competentes para responder el cuestionario propuesto, se construyó una lista de posibles encuestados para asegurar que los participantes habían trabajado para empresas de software y tenía conocimientos en estimación de esfuerzo. En segundo lugar, se recopilaron datos demográficos específicos sobre las áreas de negocio en las que trabajan los encuestados, su experiencia profesional, sus funciones y responsabilidades dentro de los equipos de software y su nivel de experiencia en la estimación de software. Todo esto fue hecho para garantizar que las respuestas dadas por los encuestados fueran confiables y adecuadas. En tercer lugar, se llevó a cabo un experimento piloto con estudiantes de posgrado y se obtuvo realimentación sobre el cuestionario en cuanto a su dificultad y el tiempo necesario para responder a las preguntas. Con base en esta información, se confirmó que el tiempo requerido para responder el cuestionario no sería superior a 15

minutos y que todas las preguntas incluidas eran claras, concisas y podían ser contestadas sin mayor dificultad.

La validez de constructo se refiere a la medida en que una encuesta mide lo que dice que mide. Con respecto a este tipo de validez, se mitigaron las amenazas asociadas diseñando el cuestionario con base en los resultados de la revisión sistemática de literatura (RSL).

Además, se garantizó la pertinencia de cada pregunta incluida asociándola con al menos una de las preguntas de investigación formuladas en este estudio. De igual forma, los elementos del cuestionario se mejoraron iterativamente para garantizar que las preguntas fueran válidas, claras, inequívocas y libres de prejuicios. Por último, se desarrolló una introducción a la encuesta donde se explicó el propósito general del estudio, se aclaró el concepto de estimación del esfuerzo y se proporcionó a los participantes información de contacto para que pudieran formular preguntas o expresar sus comentarios sobre la encuesta.

4. Capítulo 4: Conclusiones y trabajo futuro

Se desarrolló un estudio de encuesta destinado a determinar cómo los profesionales del desarrollo de software colombianos llevan a cabo el proceso de estimación del esfuerzo dentro de proyectos ágiles. La encuesta diseñada tiene 24 preguntas que indagan sobre los métodos de estimación de esfuerzo utilizados, la precisión de las estimaciones, los factores que influyen en la precisión de las estimaciones, si se utilizan *datasets* para respaldar los procesos de estimación de cualquier forma, además de la información demográfica de la población encuestada.
La encuesta fue administrada a través de una herramienta de cuestionario en línea y respondido por 60 personas que desempeñan diversos roles dentro empresas colombianas de desarrollo de software. Se usó muestreo de conveniencia, por lo tanto, la lista de participantes en este estudio se construyó a partir de los contactos de algunos miembros del grupo de investigación. De esta manera, se aseguró que todos los participantes trabajaran en la industria de software al momento de realizar el estudio.

Los resultados indican que, con respecto a los métodos de estimación, las técnicas basadas en el Juicio de Expertos son las más usadas. Dentro de esta categoría destaca Planning Poker porque es utilizado por más de la mitad de los encuestados y su efectividad se considera media-alta al estimar duración, costos y recursos de los proyectos ASD.
Además, el uso de técnicas paramétricas es representativo (regresión, COCOMO, Análisis de Puntos de Función). Una minoría de los encuestados informa que usa métodos orientados al aprendizaje y el uso de técnicas propias de la organización no es representativo.

En cuanto a los procesos ágiles utilizados, los resultados indican que Scrum ha sido utilizado por todos los participantes, mientras que Kanban y XP fueron reportados por menos de la mitad de los encuestados. En este sentido, el predominio de Scrum lo convierte en el marco típico donde tiene lugar la aplicación de técnicas de estimación del esfuerzo. Además, la estimación del esfuerzo toma lugar principalmente dentro del proceso de planificación de granularidad fina, es decir, en la historia de usuario o nivel de caso de

uso. Los datos también sugieren que los resultados de la estimación se utilizan, en menor medida, en la planificación de iteraciones, *release* y todo el proyecto en general.
Los resultados son consistentes con los estudios que describen la forma en que se planifican y gestionan los proyectos ágiles, como lo muestra la investigación realizada por Usman, Mendes y Börstler [17].

En relación con las actividades para las que tiene sentido estimar el esfuerzo, se encontró que los practicantes de ASD realizan estimación del esfuerzo principalmente para actividades de desarrollo (88,3 %), actividades de análisis (60 %), tareas de diseño (53,3 %) y trabajos de pruebas de software (51,67%).

Con respecto a las técnicas de dimensionamiento de software, los resultados indican que los practicantes de ASD usan Story Points y Use Case Points en gran medida como métodos de medición del tamaño del software. Aproximadamente un tercio de la población encuestada utiliza Puntos de Función, y Líneas de código en menor medida. Además, alrededor del 17% de los encuestados informaron que no usan técnicas de medición de tamaño.
En cuanto a las herramientas de apoyo, el uso de herramientas de software especializadas para realizar la estimación de esfuerzo no es común. Esta actividad es apoyada por hojas de cálculo (90%) y, en menor medida, por herramientas de software adaptadas a las necesidades de cada organización (que puede estar basado en hojas de cálculo), al igual que herramientas de software gratuito.

En cuanto a la precisión de la estimación, se considera alta para poco más de la quinta parte de los encuestados, media para algo más de la mitad de ellos, y bajo o indeterminado para el resto de los encuestados. De esta última proporción, sólo el 11,67% consideran sus estimaciones con un nivel de precisión bajo o muy bajo. Sin embargo, al preguntar sobre técnicas específicas para analizar el nivel de precisión de las estimaciones, alrededor del 70% de los encuestados afirman que no se lleva a cabo el proceso de análisis formal de precisión. El uso del análisis de variabilidad fue reportado por 18,3% de las personas, mientras que MRE y PRED(X) fueron reportados por una pequeña proporción de los encuestados.
Todos los factores que afectan la precisión de las estimaciones planteados en la encuesta se reconocen como significativos. Los más relevantes son la gestión de requisitos y el

equipo de proyecto en lo que a tamaño y experiencia respecta. Con respecto a los predictores de esfuerzo, los más relevantes en orden de importancia son la experiencia del equipo, la complejidad del software, tipo de desarrollo (nuevo desarrollo, mantenimiento), tamaño del equipo, dominio del problema y técnica de medición del tamaño del software.

Con respecto a los *datasets*, se encontró que, en la mayoría de los casos, no se utilizan para respaldar el proceso de estimación del esfuerzo (85%). Esto quiere decir que probablemente la velocidad sólo se está calculando a nivel de cada iteración y se va ajustando, pero no se calcula a nivel de proyecto haciendo uso de datos de proyectos pasados. Esto está en línea con el hallazgo en el que la metodología más usada es SCRUM y no Kanban, en donde el rendimiento (velocidad) sí se basa en datos reales. Por otro lado, este es un indicador del bajo nivel de uso de técnicas paramétricas u orientadas al aprendizaje, que requieren datos para calcular estimaciones o entrenar los modelos. En los pocos casos en que se utilizan *dataset*s, en su mayoría son fuentes públicas o datos históricos de la propia organización. En solo un par de casos, el uso de *datasets* se debe a algún tipo de licencia adquirida para usarlos. Los *datasets* se utilizan en gran medida para hacer comparaciones de productividad entre proyectos dentro de la propia organización (77,8%); en segundo lugar, se utilizan como insumo para la aplicación de técnicas de estimación del esfuerzo; por último, para comparar la productividad con proyectos en el software industria en general.

Con respecto a las similitudes o diferencias que se presentan entre Colombia y otros países, se puede concluir que, tanto en Colombia como en otras partes del mundo, los métodos de estimación más utilizados para realizar una estimación de esfuerzo son el Juicio de Expertos y Planning Poker. En algunos sectores acuden más a uno que a otro, pero en todos los escenarios, son estos dos los que ocupan los dos primeros lugares dentro de la amplia lista de métodos posibles a utilizar. Dado que Planning Poker acude de igual forma al Juicio de Expertos para asignar Story Points a cada historia de usuario, se puede concluir que el método de estimación más utilizado mundialmente sigue siendo el Juicio de Expertos en todas sus variaciones posibles. Otro aspecto que es posible generalizar, tiene que ver con los factores que se consideran determinantes para estimar esfuerzo. Tanto en Colombia como en el resto del mundo, de acuerdo con lo reportado por

los trabajos relacionados, el factor más importante a considerar es la experiencia del equipo del proyecto, seguido de la complejidad del software que se requiere desarrollar.

Los hallazgos de este estudio son significativos porque conocer las prácticas actuales en la industria es esencial para que los investigadores puedan determinar exactamente qué lagunas deben llenarse, al diseñar nuevas técnicas de estimación. Es más, este estudio muestra que el seguimiento de la precisión de las estimaciones a lo largo del tiempo, así como la comprensión y el control de los factores básicos que influyen en el esfuerzo del software son tareas excepcionalmente fundamentales en estimación de proyectos de software.

Para trabajos futuros, es posible realizar estudios que comparen la efectividad y precisión de los métodos basados en el aprendizaje *(machine learning - ML)* con respecto a los métodos tradicionales basados en el Juicio de Expertos. De igual forma se pueden estudiar métodos que incluyan el uso de *datasets* y su eficacia en el cálculo de la velocidad.
Por otro lado, dado que este estudio se realizó con una muestra de profesionales con experiencia en estimación de esfuerzo, se puede realizar un trabajo futuro para medir la percepción de las personas que contratan equipos con respecto a la precisión de las estimaciones.

Anexos

Anexo1 - Cuestionario en español

Estimación de esfuerzo en proyectos Ágiles de Desarrollo de Software (ENCUESTA)

INFORMACIÓN DEMOGRÁFICA

1. Seleccione el Rol en el cual usted tiene mayor experiencia: *
   - o Arquitecto de Software
   - o Analista de sistemas
   - o Diseñador de software
   - o Desarrollador de software
   - o Otro: _____

2. Rango de edad: *
   - o Menos de 25
   - o Entre 25 y 30
   - o Entre 31 y 35
   - o Entre 36 y 45
   - o Más de 45

3. Sector al que pertenece la organización para la que usted trabaja: *
   - o Financiero/Banca
   - o Gobierno
   - o Construcción y obras civiles
   - o Seguros
   - o Educación
   - o Tecnologías de la información
   - o Salud
   - o Manufactura
   - o Comercio
   - o Transporte y logística
   - o Otro: _____

4. Región geográfica desde donde usted trabaja: *
   - o Región Andina
   - o Costa Atlántica
   - o Orinoquía
   - o Pacífico
   - o Región Insular
   - o Amazonía
   - o Otra: _____

5. ¿Qué tan larga ha sido su experiencia en la industria de desarrollo de software? *
   - o Menos de 1 año
   - o Entre 1 y 2 años

- Más de 2 y menos de 5 años
- Más de 5 años

6. ¿Qué tan larga ha sido su experiencia realizando estimación de esfuerzo en proyectos de software? *
   - Ninguna experiencia
   - Menos de 1 año
   - Entre 1 y 2 años
   - Más de 2 y menos de 5 años
   - Más de 5 años

7. Seleccione el último nivel académico alcanzado: *
   - Técnico / Tecnológico
   - Pregrado
   - Especialización
   - Maestría
   - Doctorado

MÉTODOS DE ESTIMACIÓN DE ESFUERZO

8. Seleccione las técnicas de estimación de esfuerzo que ha utilizado en sus proyectos de desarrollo ágil. Seleccione todas las que apliquen. *
   - □ Juicio de Expertos - Delphi
   - □ Juicio de Expertos - Wideband Delphi
   - □ Planning Poker
   - □ Analogía
   - □ Top-Down
   - □ Bottom-Up
   - □ Regresión
   - □ COCOMO II (Constructive Cost Model)
   - □ Orientados al aprendizaje (Machine Learning)
   - □ Otros: _____

9. Seleccione los métodos / Frameworks que ha utilizado en proyectos ágiles en los que ha participado. Seleccione todos los que apliquen. *
   - □ Scrum
   - □ Kanban
   - □ Extreme Programming (XP)
   - □ Crystal
   - □ Lean Software Development
   - □ Dynamic Systems Development Method (DSDM)
   - □ Test Driven Development (TDD)
   - □ Feature Driven Development (FDD)
   - □ Adaptive Software Development
   - □ SAFe

□ Otros: _____

10. Seleccione para cuáles niveles de planeación usted ha realizado estimación de esfuerzo. Seleccione todos los que apliquen. *
    □ Proyecto
    □ Release
    □ Iteración
    □ Historia de usuario / Requerimiento / Caso de Uso
    □ Ninguno
    □ Otro: _____

11. ¿Para cuáles de las siguientes actividades realiza estimación de esfuerzo? Seleccione todas las que apliquen. *
    □ Actividades de análisis y requerimientos
    □ Actividades de diseño
    □ Actividades de desarrollo/construcción
    □ Actividades de pruebas
    □ Actividades de mantenimiento
    □ Actividades de documentación
    □ Actividades de entrenamiento
    □ Otras: _____

12. Seleccione las herramientas que utiliza para apoyar el proceso de estimación de esfuerzo. Seleccione todas las que apliquen. *
    □ Hojas de cálculo
    □ Herramientas de software gratuitas (Ejemplo: USC COCOMO, COCOMO Online)
    □ Herramientas de software licenciadas (Ejemplo: SEER, SLiM, COSTAR y PRICE-S)
    □ Herramientas de software hechas a la medida
    □ Ninguna
    □ Otras: _____

13. Seleccione cuáles técnicas de medición de tamaño de software ha utilizado en sus proyectos de desarrollo ágil. Seleccione todas las que apliquen. *
    □ Story Points
    □ Function Points (IFPUG, COSMIC Points, etc.)
    □ Líneas de código
    □ Use Case Points
    □ Object Points
    □ Ninguna
    □ Otras: _____

EXACTITUD EN LA ESTIMACIÓN

14. En general, para los proyectos en los que ha participado, ¿Cuál ha sido el nivel de precisión de las estimaciones de esfuerzo? *
    ○ No se valida el nivel de precisión
    ○ Muy bajo
    ○ Bajo
    ○ Medio
    ○ Alto
    ○ Muy alto

15. Seleccione las técnicas utilizadas para analizar el nivel de exactitud de las estimaciones. Seleccione todas las que apliquen. *
    □ No se realiza análisis de exactitud de las estimaciones
    □ PRED (x) - Percentage Relative Error Deviation

□ MRE - Magnitude Relative Error
□ Análisis de Variabilidad
□ Otro: _____

16. De los siguientes aspectos ¿Cuáles afectan la precisión de la estimación? Seleccione todos los que apliquen. *
□ Requerimientos (Definición, especificación, validación)
□ Gestión del proyecto
□ Equipo de proyecto (Tamaño, experiencia)
□ Optimismo/Pesimismo desmedido
□ Formalidad en el proceso de estimación
□ Involucramiento del cliente
□ Otro: _____

17. ¿Ha utilizado Planning Poker como técnica de estimación?
○ Si
○ No

EXACTITUD EN LA ESTIMACIÓN – PLANNING POKER

18. Con relación a Planning Poker, señale que tan efectiva ha sido su utilización para estimar los siguientes aspectos del proyecto:

	Muy baja	Baja	Media	Alta	Muy alta	No Aplica
Duración	○	○	○	○	○	○
Costo	○	○	○	○	○	○
Recursos	○	○	○	○	○	○

PREDICTORES DE ESFUERZO

19. Seleccione las variables que considera fundamentales para apoyar el proceso de estimación de esfuerzo. Seleccione todos los que apliquen. *
□ Experiencia del equipo
□ Tamaño del equipo
□ Lenguaje de programación
□ Dominio del software (Web, Cliente servidor, mainframe, software embebido, etc.)
□ Tipo de desarrollo (Nuevo desarrollo, mantenimiento)
□ Método de desarrollo ágil (SCRUM, XP, Kanban, FDD, TDD, BDD, etc.)
□ Desarrollo in-house / Outsourcing
□ Complejidad del software
□ Técnica de medición de tamaño (Function Points, Story Points, Use Case Points, etc.)
□ Duración del proyecto
□ Otra: _____

20. ¿Cuáles de las siguientes variables ha tenido en cuenta para realizar el proceso de estimación de esfuerzo en sus proyectos? Seleccione todos los que apliquen. *

□ Experiencia del equipo

□ Tamaño del equipo
□ Lenguaje de programación
□ Dominio del software (Web, Cliente servidor, mainframe, software embebido, etc.)
□ Tipo de desarrollo (Nuevo desarrollo, mantenimiento)
□ Método de desarrollo ágil (SCRUM, XP, Kanban, FDD, TDD, BDD, etc.)
□ Desarrollo in-house / Outsourcing
□ Complejidad del software
□ Técnica de medición de tamaño (Function Points, Story Points, Use Case Points, etc.)
□ Duración del proyecto
□ Otra: _____

DATASETS PARA ESTIMACIÓN

21. ¿Ha utilizado datasets para apoyar el proceso de estimación de esfuerzo? *
    - o Si
    - o No

22. Seleccione datasets que utiliza o ha utilizado para apoyar el proceso de estimación de esfuerzo. Seleccione todos los que apliquen. *
    - □ Datasets públicos (de acceso gratuito)
    - □ Datasets licenciados (requieren la adquisición de una licencia)
    - □ Datasets propios de la organización (histórico de proyectos de la empresa)
    - □ Otros: _____

23. Seleccione la forma en que ha utilizado los datasets. Seleccione todas las que apliquen. *
    - □ Aplicando técnicas de estimación de esfuerzo sobre el dataset
    - □ Realizando comparaciones de productividad entre proyectos de la organización
    - □ Realizando comparaciones con productividades de industria
    - □ Otros: _____

Observaciones

Por favor déjenos sus comentarios con respecto a esta encuesta.

_____

Referencias

[1] M. Chemuturi, Software Estimation Best Practices & Techniques, Broward-Florida: J.Ross Publishing, 2009.

[2] I. Project Management Institute, A Guide to the Project Management Body of Knowledge (PMBOOK), Newtown Square-Pennsylvania USA: PMI Publishing Division, 2017.

[3] I. Sommerville, Software Engineering - Global Edition, 10th ed.Harlow, England: Pearson Education, 2016.

[4] M. Qureshi, "Agile Software Development Methodology for Medium and Large Projects," *IET Software,* vol. 6, no. 4, pp. 358-363, 2012.

[5] O. Malgonde and K. Chari, "An Ensemble-Based Model for Predicting Agile Software Development Effort," *Empirical Software Engineering,* vol. 24, no. 2, pp. 1017-1055, 2020.

[6] M. Fernandez-Diego, E. Mendez, F. Gonzalez, S. Abrahão and E. Insfran, "An Update on Effort Estimation in Agile Software Development: A Systematic Letreature Review," *IEE Access,* vol. 8, pp. 166768-166800, 2020.

[7] E. Ruel, W. E. Wagner III and B. J. Gillespie, The Practice of Survey Research, SAGE Publications, Inc., 2016.

[8] B. Kitchenham and S. Pfleeger, "Principles of Survey Research. Part 1," *Software Engineering Notes,* vol. 26, no. 1, pp. 16-18, 2001.

[9] J. Floyd J. Fowler, Survey Research Methods. Fifth Edition, Boston: SAGE Publications, Inc., 2014.

[10] C. Wohlin, P. Runeson, M. H¨ost, M. C. Ohlsson and B. Regnell, Experimentation in software engineering, Springer, 2012.

[11] J. Linåker, S. M. Sulaman, R. Maiani de Mello and M. Höst, "Guidelines for Conducting Surveys in Software Engineering," Department of Computer Science, Lund University, 2015.

[12] J. Klagge, "Guidelines for Conducting Surveys," Arizona, 2018.

[13] S. Zia and S. Ziauddin, "An Effort Estimation Model for Agile Software Development," *Adv. Comput. Sci. its Appl. (ACSA),,* vol. 314, no. 1, pp. 2166-2924, 2012.

[14] T. Dingsøyr, T. Dybå and N. B. Moe, Agile Software Development: An Introduction and Overview, Heidelberg ,Germany: Springer, 2010.

[15] "Agile Manifesto for Software Development—Agile Alliance," [Online]. Available: https://www.agilealliance.org/agile101/the-agilemanifesto. [Accessed 5 Oct 2021].

[16] L. Williams and A. Cockburn, "Guest Editors' Introduction: Agile Software Development: It's About Feedback and Change," *Computer,* vol. 36, no. 6, p. 39–43, 2003.

[17] M. Usman, E. Mendes and J. Börstler, "Effort Estimation in Agile Software Development: A Survey on the State of the Practice," in *19th International Conference on Evaluation and Assessment in Software Engineering (EASE)*, Nanjing, China, 2015.

[18] "scrumalliance.org," [Online]. Available: https://scrumalliance.org. [Accessed 15 02 2022].

[19] Ashmore, Sondra and K. Runyan, Introduction to Agile Methods, Addison-Wesley Professional, 2014.

[20] A. Singh, Agile & Scrum, Babelcube, Inc, 2021.

[21] "https://www.agilealliance.org/," .agilealliance.org, 15 01 2022. [Online]. Available: https://www.agilealliance.org/. [Accessed 2022].

[22] M. Poppendieck and T. Poppendieck, Lean Software Development: An Agile Toolkit, Addison-Wesley, 2003, 2003.

[23] P. Abrahamsson, O. Salo, J. Ronkainen and J. Warsta, Agile Software Development Methods: Review and Analysis, 2017.

[24] M. Al-Zewairi, M. Biltawi, W. Etaiwi and A. Shaout, "Agile Software Development Methodologies:Survey of Surveys," *Journal of Computer and Communications,* vol. 5, pp. 74-97, 2017.

[25] C. G. Cobb, The Project Manager's Guide to Mastering Agile: Principles and Practices for an Adaptive Approach, Wiley, 2015.

[26] "https://www.scaledagileframework.com/," 21 04 2022. [Online]. Available: https://www.scaledagileframework.com/.

[27] P. M. Institute, Practice Standard for Project Estimating, Pennsylvania: Newtown Square, 2019.

[28] J. Wen, S. Li, Z. Lin, Y. Hu and C. Huang, "Systematic Literature Review of Machine Learning Based Software Development Effort Estimation Models," *Inf. Softw. Technol.,* vol. 54, no. 1, pp. 41-49, 2012.

[29] A. Trendowicz, Software Project Effort Estimation, Berlin, Germany: Springer-Verlag, 2014.

[30] P. Abrahamsson, R. Moser, W. Pedrycz, A. Sillitti, Succi and G., "Effort Prediction in Iterative Software Development Processes - Incremental Versus Global Prediction Models," in *1st International Symposium on Empirical Software Engineering and Measurement*, 2007.

[31] M. Cohn, Agile Estimating and Planning, Publisher: Prentice Hall, 2005.

[32] F. Tsui, O. Karam and B. Bernal, Essentials of Software Engineering, 5th Edition, Jones & Bartlett Learning, 2022.

[33] S. McConnell, Software Estimation: Demystifying the Black Art, Addison-Wesley Professional, 2006.

[34] M. A. Parthasarathy, Practical Software Estimation: Function Point Methods for Insourced and Outsourced Projects, Addison-Wesley Professional, 2007.

[35] B. Boehm, J. A. Lane, S. Koolmanojwong and R. Turner, The Incremental Commitment Spiral Model: Principles and Practices for Successful Systems and Software, Addison-Wesley Professional, 2014.

[36] M. Bisi and N. Goyal, Artificial Neural Network Applications for Software Reliability Prediction, Wiley-Scrivener, 2017.

[37] R. T. Futrell, D. F. Shafer and L. Safer, Quality Software Project Management, Pearson, 2002.

[38] R. Rishi, "Early Size Estimation using Machine Learning," in *8th International Conference on Computing for Sustainable Global Development*, Rohtak, India, 2021.

[39] P. Monika, "Software effort estimation using machine learning," in *7th International Conference on Cloud Computing, Data Science & Engineering* , Hisar, Haryana, 2017.

[40] E. T. Mueller, Commonsense Reasoning, 2nd Edition, Morgan Kaufmann, 2014.

[41] A. Stellman and J. Greene, Applied software project management, “O’Reilly, 2005.

[42] M. Jørgensen, "A Critique of How We Measure and Interpret the Accuracy of Software Development Effort Estimation," in *1st International Workshop on Software Productivity Analysis and Cost Estimation*, Nagoya, Japan, 2007.

[43] D. Port and M. Korte, "Comparative Studies of the Model Evaluation Criterions MMRE and PRED in Software Cost Estimation Research.," in *Proceedings of the Second International Symposium on Empirical Software Engineering and Measurement*, Kaiserslautern, Germany, 2008.

[44] M. Usman, E. Mendes, F. Weidt and R. Britto, "Effort Estimation in Agile Software Development: A systematic literature review," in *International Conference on Predictive Models in Software Engineering - PROMISE 2014*, Turin, Italy, 2014.

[45] T. Vera, S. Ochoa and D. Perovich, "Understanding the Software Development Effort Estimation in Chilean Small Enterprises," University of Chile, Santiago de Chile, 2018.

[46] Z. Prokopova, P. Silhavy and R. Silhavy, "Analysis of the Software Project Estimation Process: A Case Study," *Software Engineering Methods in Intelligent Algorithms,* vol. 984, pp. 456-467, 2019.

[47] M. A. Ramessur and S. Nagowah, "Factors Affecting Sprint Effort Estimation," *Advances in Intelligent Systems and Computing,* vol. 1089, pp. 507-518, 2020.

[48] F. Prieto, "Estimación de esfuerzo en desarrollo de softearw ágil: Estudio del estado actual en Bogota," *ITECKNE,* vol. 17, no. 2, p. 22, 2020.

[49] M. Majchrzak and L. Madeyski, "Factors influencing User Story estimations: an industrial interview and a conceptual model," *Cent. East. Eur. J. Manag. Econ,* vol. 4, no. 4, pp. 261-280, 2016.

[50] P. Abrahamsson, I. Fronza, R. Moser, J. Vlasenko and W. Pedrycz, "Predicting Development Effort from User Stories," in *International Symposium on Empirical Software Engineering and Measurement*, Banff, AB, Canada, 2011.

[51] V. Mahniˇc and T. Hovelja, "On Using Planning Poker for Estimating User Stories," *Journal of Systems and Software,* vol. 85, no. 9, pp. 2086-2095, 2012.

[52] N. C. Haugen, "An Empirical Study of Using Planning Poker for User Story Estimation," in *Conf AGILE 2006 (AGILE'06)*, Minneapolis, MN, USA, 2006.

[53] Colombia-MINTIC, "https://colombiatic.mintic.gov.co," Enero 2015. [Online]. Available: https://colombiatic.mintic.gov.co/679/w3-article-73973.html. [Accessed Agosto 2021].