UNIVERSIDAD NACIONAL DE COLOMBIA

Un caso de estudio sobre los factores que afectan la estimación de proyectos de software con base en el repositorio ISBSG

Claudia Milena Sabogal Serrano

Universidad Nacional de Colombia
Facultad de Minas, Departamento de Ciencias de la Computación y la Decisión
Medellín, Colombia
2017

Un caso de estudio sobre los factores que afectan la estimación de proyectos de software con base en el repositorio ISBSG

Claudia Milena Sabogal Serrano

Tesis de grado presentada como requisito parcial para optar al título de:
Magister en Ingeniería – Ingeniería de Sistemas

Director:
Albeiro Espinosa Bedoya, PhD.

Línea de Investigación:
Ingeniería del Software
Grupo de Investigación:
Calidad del Software

Universidad Nacional de Colombia
Facultad de Minas, Departamento de Ciencias de la Computación y la Decisión
Medellín, Colombia
2017

A mi amada hijita, Ámbar.

A mi compañero de vida, Andrés.

A mis excelentes padres, Ramiro y Claudia.

A mis lindas hermanitas, Luciana y Valentina.

Agradecimientos

Principalmente e infinitos agradecimientos, a mi Director de tesis, Profesor Albeiro Espinosa Bedoya por guiarme con sus conocimientos en este reto académico tan importante para mí. Por el compromiso, la paciencia y la motivación que siempre me brindó hasta lograr culminar de manera satisfactoria el presente trabajo.

A mi preciosa Ámbar, por ser mi persona favorita, la luz de mi vida y mi motivación. Todo es por y para ti.

A mi compañero de vida, por creer en mí, por el apoyo y la paciencia.

A mis padres y hermanitas, por el amor, los valores familiares y el apoyo incondicional.

A todas las personas; profesores, compañeros de estudio y de trabajo y amigos que me acompañaron en este proceso académico y personal.

A la Universidad Nacional de Colombia, Sede Medellín, Facultad de Minas, por permitir y contribuir con mí desarrollo académico, laboral y personal mediante incentivos tan valiosos como la beca de estudio para empleados, permisos de estudio y la disposición de los mejores recursos académicos.

Resumen

La estimación del esfuerzo en el desarrollo de software es una tarea compleja, que involucra múltiples factores que la afectan, tanto metodológicos como organizacionales y que se traducen en incertidumbre e imprevisibilidad en la estimación, fracaso de los proyectos por subestimación o sobreestimación y en general pérdidas de dinero. Este problema lo viene abordando la comunidad del área empleando enfoques como las redes neuronales, el juicio de expertos, entre otras. En este Trabajo Final de Maestría se desarrolla un caso de estudio sobre los factores que afectan la estimación de proyectos de software con base en el repositorio ISBSG. El análisis de estos factores permitió identificar como los más importantes a: la planeación, la especificación, el diseño, la construcción, las pruebas, la implementación, el tipo de desarrollo, el tamaño del proyecto, el método de estimación del esfuerzo, el nivel de recursos y el tamaño del grupo.

Palabras clave: estimación, esfuerzo, software, costo, proyecto

Abstract

Effort estimation in software development is a complex task, involving multiple factors that affect it, both methodological and organizational and that results in uncertainty and unpredictability in the estimation, failure of the projects due to underestimation or overestimation and, in general, losses of money. This problem is being addressed by the community of the area using approaches such as neural networks, expert judgment, among others. In this Final Master's Project develops a case study on the factors that affect the estimation of software projects based on the ISBSG repository. The analysis of these factors allowed identifying as the most important factors: planning, specification, design, construction, testing, implementation, type of development, project size, effort estimation method, level of resources and group size.

Keywords: Estimation, effort, software, cost, project.

Contenido

Lista de figuras

Pág.

Lista de tablas

Pág.

1. Introducción

1.1 Introducción

En la ingeniería de software, el esfuerzo es la cantidad de trabajo que un equipo de desarrollo necesita para culminar las actividades específicas dentro de un proyecto desarrollo de software, generalmente calculado en persona por unidad de tiempo (Pushkar y Kumari, 2013; Trendowicz y Jeffery, 2014). La estimación del esfuerzo en proyectos de desarrollo de software es importante ya que permite clasificar y priorizar los proyectos de acuerdo con un plan de negocios general, definir y asignar en etapas tempranas del proyecto los recursos humanos y técnicos que se van a comprometer con el mismo y prevenir riesgos. En general, es más fácil administrar y controlar un proyecto con recursos adaptados a las necesidades reales siendo los resultados más consistentes con lo planeado, lo que permite desarrollar y completar el proyecto bajo los términos de tiempo, recursos y funcionalidad establecidos (Aparna *et al*., 2015; Mansor *et al*., 2011; Pushkar y Kumari, 2013; Rajkumar y Alagarsamy, 2013).

La estimación del esfuerzo en el desarrollo de software es una tarea difícil. El riesgo y la incertidumbre son altos y la confianza en la estimación es baja. Los procesos de desarrollo de software involucran numerosas fuentes de variabilidad como las humanas, técnicas y de entorno organizacional que contribuyen a la imprecisión de la estimación del costo final del software y el esfuerzo para desarrollarlo, impactando directamente la calidad de los proyectos (Dejaeger, Verbeke, Martens y Baesens, 2012; Pushkar y Kumari, 2013; Tailor, Saini y Rijwani, 2014).

Desde los años 60s hasta la fecha, tanto la comunidad académica como la empresarial estudian y desarrollan modelos de estimación, con el objetivo general de controlar los costos e incrementar los niveles de servicio y calidad del software. Existen numerosas técnicas de estimación que se pueden clasificar en dos tipos: modelos algorítmicos y modelos no algorítmicos (Pushkar y Kumari, 2013). Los primeros se basan en el uso de relaciones matemáticas y difícilmente producen estimaciones certeras en situaciones especiales con pocos datos de entrada; los segundos se basan en comparaciones e inferencias analíticas, generan estimaciones rápidamente, se adaptan a situaciones especiales y dependen de personas expertas (Aparna *et al.*, 2015). En la actualidad, en la

búsqueda de mejorar la precisión de la estimación siguen surgiendo modelos que combinan los diferentes métodos y técnicas de estimación (Robiolo y Santos, 2013; Macdonell y Shepperd, 2003).

La industria de TI (Tecnologías de la Información) desarrolló dos bases de datos internacionales con datos históricos de estimaciones de proyectos reales que sirven como base para la estimación de proyectos similares. Estas son: ISBSG *(International Software Benchmarking Standars Group)* y CSBSG *(Chinese Software Benchmarking Standards Group)*. Las pequeñas empresa de desarrollo de software conforman la industria de TI de la mayoría de los países. Estas favorecen el crecimiento de las economías nacionales mediante el desarrollo de productos significativos y requieren prácticas eficientes de ingeniería de software adaptadas a su tamaño y tipo de negocio (Páez, 2012; Pino, García, y Piattini, 2006).

En casos de estudio como: Malasia (Mansor, Kasirun, Yahya, y Arshad, 2011), Pakistán (Nasir y Ahmad, 2006), China (Yang *et al.*, 2008), Noruega (Moloekken-OEstvold *et al.*, 2004), Venezuela (Villareal, 2007) y Colombia (Páez, 2012), se evidencia que la industria local de desarrollo de software recibe poca atención en su proceso de estimación de esfuerzos de desarrollo. En el contexto local, Colombia, las pequeñas empresas dominan la industria del software y generalmente no utilizan los modelos estándar de estimación de proyectos, ya que se desconfía de estos porque no representan la realidad y las particularidades de estas organizaciones y del entorno en que desarrollan sus actividades (Páez, 2012).

En estas empresas existe aún incertidumbre sobre cómo valorar, implementar y evaluar la precisión de los diferentes métodos de estimación y predecir estimaciones certeras de los esfuerzos y costos de desarrollo de software (Páez, 2012; Pino, García, y Piattini, 2006). Por tanto, se emplea con mayor frecuencia el método de estimación por Juicio de Experto porque puede hacer predicciones rápidas y más ajustadas a la realidad y tener en cuenta particularidades de la organización pero, se asume el riesgo de la dependencia de este o de no poder contar con él (Páez, 2012).

Por lo anterior, en este trabajo se desarrolla un caso de estudio sobre los factores que afectan la estimación de proyectos de software con base en el repositorio ISBSG.

1.2 Motivación

A pesar del amplio abordaje y la amplia experimentación con modelos de estimación tanto de la academia como de industria del software, no se tiene un consenso en el proceso de estimación de esfuerzo que permita mejorar la precisión de las predicciones. La estimación del esfuerzo de desarrollo de software se traduce en un asunto de dinero, que el tiempo y la premura en obtener el producto afectan.

En América Latina y de manera específica en Colombia, las pequeñas empresas dominan la industria del software (Peláez Valencia, *et al*, 2011). Estas empresas se dedican al desarrollo de software a la medida, denominado software artesanal, es decir, adecuado especialmente a las necesidades del cliente, por tanto, no utilizan métodos de desarrollo existentes en el contexto internacional (Peláez Valencia, *et al*, 2011). Estos son proyectos generalmente de pequeño tamaño, expuestos a la falta de tiempo, la incertidumbre y la imprevisibilidad dando como resultados proyectos que fracasan por la subestimación o la sobreestimación de los mismos (Páez, 2012; Pino, García, y Piattini, 2006, Salvetto De León, 2006).

La estimación de esfuerzo, como una de las causas de fracaso en los proyectos de desarrollo de software de tamaño pequeño y mediano (Robiolo y Santos, 2014), es el interés de este trabajo. La industria local de desarrollo de software se puede beneficiar en términos de costos y disminución del fracaso de proyectos con el análisis, identificación y caracterización de los factores de mayor incidencia en un modelo de estimación ajustado a sus necesidades.

1.3 Planteamiento del Problema

La estimación de esfuerzos en proyectos de desarrollo software es una tarea compleja. El proceso involucra factores inherentes a la naturaleza del software como su intangibilidad y dinamismo, que dificultan la valoración de un producto no visible y que, además, cambia constantemente de requisitos, afectando la precisión y replicabilidad del proceso (Dejaeger, Verbeke, Martens, y Baesens, 2012; Pushkar y Kumari, 2013; Tailor, Saini y Rijwani, 2014).

Este proceso también involucra factores propios del ciclo de vida desarrollo y de la experiencia de la organización como: el tipo de proyecto, el tamaño, la disponibilidad de recursos e información histórica, entre otras, que impactan la precisión de las estimaciones y la productividad de los proyectos (Pushkar y Kumari, 2013). Es, entonces, una tarea que requiere disponer de información detallada del proyecto a estimar, realizar una buena planificación del proyecto y conocer los recursos disponibles (Ruíz Constanten y Cordero Morales, 2013).

Sin embargo, en la estimación de proyectos de software de tamaño pequeño y mediano, la industria local no dispone de los recursos y del conocimiento de dichos factores, por lo que las estimaciones de los proyectos de software suelen errar y pronosticar resultados sobreestimados o subestimados frente al resultado real. Esta mala planeación y/o ejecución de los proyecto de software causa pérdidas traducidas principalmente en costos y tiempo.

Por lo anterior, en este trabajo se desarrolla un caso de estudio sobre los factores que afectan la estimación de proyectos de software con base en el repositorio ISBSG.

1.4 Trabajo Previo

Los investigadores de la ingeniería del software, pioneros en el estudio de modelos de estimación que permitan asignar el presupuesto, analizar los riesgos, planificar los proyectos, analizar las inversiones y controlar y mejorar el software (Ayyıldız, Kalıpsız, y Yavuz, 2006; Cerpa y Verner, 2009; Pushkar y Kumari, 2013; Rajkumar y Alagarsamy, 2013), son:

Lawrence H. Putnam, con su modelo algorítmico para la estimación de costos de software llamado SLIM (*Software Lifecycle Management*); (Putnam, 1978); Alan Albrecht de IBM, con su modelo FPA (*Function Point Analysis*) usado también como métrica para medir el tamaño del software basada en la funcionalidad del mismo (Albrecht, 1979); Barry Boehm, con su modelo algorítmico de estimación de costos llamado COCOMO (*Constructive Cost Model*), en sus diferentes versiones (Boehm, 1981; Boehm, 2000a; Boehm *et al*. 2000b). En la actualidad, en la búsqueda de mejorar la precisión de la estimación, siguen surgiendo numerosos modelos que combinan los tradicionales métodos y técnicas de estimación con nuevas técnicas de la inteligencia artificial como la lógica difusa, sistemas basados en conocimiento y redes neuronales (Ruíz Constanten y Cordero Morales, 2013).

1.4.1 Basados en Juicio de Expertos

En el periodo de posguerra de la Segunda Guerra Mundial, el uso de la opinión de expertos en las ciencias y sus aplicaciones se convirtió en algo común. En 1946 con el “proyecto RAND” se crearon dos métodos para el Juicio estructurado de expertos: el análisis de escenarios y el método Delphi. Posteriormente en 1991 se introdujo el modelo clásico de juicio estructurado de expertos. Su uso sigue creciendo desde finales del siglo pasado hasta la actualidad (Morales y Cooke, 2010).

El aspecto clave en el cual se involucra el juicio de expertos es la incertidumbre inherente en temas de pronóstico y estimación. La cantidad y variabilidad de los factores que se involucran en un proyecto software hacen que su incertidumbre sea alta y, en consecuencia, propicia para la opinión del experto. En el dominio del desarrollo de software típicamente los expertos tienen más información específica del contexto que los modelos. por lo cual en situaciones inestables sus predicciones son mejores (Santos, 2014). El experto posee el conocimiento de proyectos previos, lo que se

constituye en una experiencia base para emitir sus conceptos acerca de nuevos proyectos. De igual forma, pueden poner en contexto información cualitativa difícil de incorporar en modelos de estimación. Morales y Cooke (2010) expresan que el juicio del experto se emplea cuando los parámetros de interés de un problema no son medibles y, en cambio, existe la experiencia, teorías y mediciones relevantes al mismo.

El pronóstico y su continua mejora son importantes para la toma de decisiones organizacionales. Las empresas de software se enfrentan a la dura tarea de pronosticar y seleccionar el método de estimación más adecuado para su contexto, con el propósito de disminuir la incertidumbre entre los valores estimados y los valores reales del proyecto de software. Algunas de ellas desarrollan su propio modelo, con el fin de mejorar la precisión. Pérez, González, Duque, Millane y Ospina (2006) plantean un modelo dinámico basado en el histórico y la experiencia de proyectos ya desarrollados en la empresa ORBITEL S.A. A partir de esta información, los autores construyen parámetros, variables y funciones. El modelo generado permite estimar tempranamente, planear y simular la capacidad del recurso humano.

1.4.2 Basados en Métricas

Las métricas son conjuntos de datos que permiten evaluar aspectos críticos en la estimación de esfuerzo y costo. Algunas de las más usadas son la complejidad funcional, la conectividad y la seguridad, entre otras. Son necesarias para unificar criterios de medición que permitan planificar y controlar los proyectos en búsqueda de la mejora continua del proceso de desarrollo y calidad del producto software. En relación con las métricas Ayyildiz, Kalipsiz y Yavuz (2006) proponen un conjunto de 26 métricas denominadas YEEM, aplicables a redes neuronales artificiales y agrupadas en varias categorías (productos, recursos, riesgos, tecnología, medio ambiente, planes y predicciones), para dar apoyo en el proceso de estimación.

La estimación temprana de un proyecto de software es difícil porque en el momento inicial se tienen menos datos para realizarla y se está evaluando la factibilidad del proyecto. Para mejorar la estimación temprana, Robiolo (2009) propone un conjunto de tres unidades de medida de software denominadas: Transacciones (*Transactions*, *T*), Objetos de Entidad

(*Entity Objects, EO*) y Caminos (*Paths, P*); cada una de ellas es función de un atributo interno del software: funcionalidad, datos y complejidad, respectivamente.

Del Valle, Cueto y Navarro (2014) proponen un conjunto de métricas para calcular el costo de un producto de software y determinar la factibilidad del proyecto. Las métricas planteadas incorporan como elemento novedoso la gestión del conocimiento de los profesionales involucrados en el proyecto. Estas métricas son: cohesión del equipo de desarrollo, experiencia con la tecnología utilizada, complejidad del modelo de datos, facilidades administrativas, experiencia con el lenguaje de programación, continuidad del personal y cronograma de desarrollo requerido.

A pesar de los diferentes enfoques en el tema de métricas, se encuentra que, por la naturaleza misma del proceso de estimación (alta incertidumbre), aún se requieren estudios adicionales.

1.4.3 Basados en Estándares de Calidad

Algunos de los aspectos que dificultan el proceso de estimación son la variabilidad de los requisitos y la falta de madurez de estos al inicio del proyecto (Páez, 2012; Pino, García y Piattini, 2006). Debido a esto, la madurez de los procesos en una organización puede ser un elemento que permita mejorar la estimación (Pesado *et al*., 2014; Yahya, Ahmad y Lee, 2008).

De León (2006) propone dos indicadores de complejidad, estos son: ICEDE (indicador de complejidad de la estructura de los datos derivado del modelo de estimación de esfuerzo) e ICEDT (Ídem del modelo de estimación de tiempo). Además, propone los modelos METEICEDE (modelo de estimación temprana de esfuerzo con base al (ICEDE) y MEPMTICEDT (modelo de estimación post mortem de tiempo) con base al (ICEDT). Estos se aplican a sistemas de gestión intensiva, en etapas tempranas, desarrollados en diferentes métodos y/o herramientas, en los que la empresa pueda construir sus propios modelos de estimación basados también en su experiencia previa, mejorando la precisión de una estimación temprana de esfuerzo y tiempo.

Yahya, Ahmad y Lee (2008) proponen el desarrollo de un nuevo conjunto de indicadores de clasificación PMAT (proceso de madurez) para COCOMO II basados en CMMI *(Capability Maturity Model Integration)* que reflejen adecuadamente el impacto de la madurez en el proceso de estimación. Los resultados muestran mejoras en la precisión en un 16% para organizaciones de CMMI nivel uno, un 34% de CMMI nivel dos, un 50% de CMMI nivel tres y un 33% de CMMI Nivel cuatro.

Un aspecto que impacta la calidad del producto software es el proceso de pruebas de software, ya que tiene por objetivo reducir el número de defectos presentes en él. Desarrollar este proceso implica tener en cuenta tanto costos directos como indirectos, que inciden en la estimación final de esfuerzos y costos del proyecto. Torres Ricaurte (2013) propone un método para medir la productividad del equipo de pruebas, usando como unidades de salida el número de casos de prueba ejecutados y como unidad de entrada el esfuerzo requerido para ejecutarlo. Este método contribuye a obtener datos de las características que influyen sobre la productividad que permiten definir índices con los cuales se puedan tomar decisiones acerca del esfuerzo de pruebas.

El juicio de expertos es importante en empresas de desarrollo de pequeños proyectos software, ya que se ajusta bien a las necesidades de éstas en relación con la estimación de esfuerzo. Santos (2014) presenta el Método de Estimación Basado en la *Especificación de Requisitos* (MEBER) bajo el estándar IEEE Std 830-1998, para la estimación de proyectos de software pequeños con juicio de expertos. El método mejora la satisfacción del cliente al lograr el cumplimiento de los requisitos acordados para los proyectos, pero no logra mejorar el error relativo de la estimación.

1.4.4 Basados en Combinación de Modelos y *Soft Computing*

Los modelos tradicionales de estimación de esfuerzo tales como COCOMO, SLIM y Puntos de Función, entre otros, ampliamente usados en la industria del software, evidencian que la precisión de la estimación aún no alcanza los niveles deseados. Por ello los investigadores desarrollan nuevos modelos que combinan los métodos tradicionales entre sí y/o con técnicas como la inteligencia artificial.

Attarzadeh y Hock Ow (2010) presentan un modelo de costos basado en *Soft Computing* incorporando características de las redes neuronales como la capacidad de aprendizaje e interpretabilidad, para el manejo de la imprecisión y la incertidumbre en la estimación de costos de software aumentando la precisión. Kashyap, Shukla y Misra (2014) proponen un método que utiliza técnicas de la teoría de la lógica difusa para mejorar la exactitud del método de puntos de casos de uso hasta en un 22% comparado con COCOMO.

Mendes (2012) propone el enfoque de modelos centrados en el experto (BNs) en combinación con una técnica que permite la inclusión explícita de la incertidumbre y las relaciones causales, como medio para mejorar la estimación del esfuerzo de software, de manera específica en la adaptación de redes bayesianas. Este enfoque mejora el proceso y la precisión de la estimación del esfuerzo de desarrollo de software, siendo esto más ventajoso para las compañías.

Otros enfoques desde el *Soft Computing* combinan algoritmos heurísticos y meta heurísticos. Kaushik, Soni y Soni (2012) proponen un enfoque de aprendizaje adaptable usando redes neuronales, basado en el algoritmo *backpropagation* para la estimación de costos de software. El modelo se puede integrar al tradicional COCOMO mejorando su rendimiento. Gharehchopogh, Maleki y Reza Khaze (2014) proponen un modelo utilizando optimización por enjambre de partículas *(Particle Swarm Optimization)* para optimizar la efectividad de los parámetros o factores de estimación. Nuevamente desde el enfoque de la inteligencia de enjambres, Maleki, Ghaffari y Masdari (2014) proponen un modelo híbrido entre Algoritmos Genéticos y Algoritmos de Colonia de Hormigas (*Ant Colony Optimization*) para la optimización del peso de los factores en los proyectos de software, reduciendo la magnitud del error relativo.

Incorporando estándares de calidad del producto software en combinación con *Soft Computing*, Almache, Raura, Ruiz y Fonseca (2015) proponen un Modelo Neuronal de Estimación para el Esfuerzo de Desarrollo en Proyectos de Software (MONEPS) utilizando una red neuronal artificial en *Backpropagation*, cuya capa de entrada se basa en un conjunto de atributos de naturaleza funcional y no funcional tomados de la norma de calidad del software ISO/IEC 25000, para la mejora de la precisión en la estimación. El error relativo promedio en tiempo y costo, es respectivamente, 10 y 3 veces menor comparado con COCOMO.

A partir de la combinación de modelos tradicionales, Kaur y Singh (2015) proponen un modelo paramétrico híbrido entre COCOMO, SLIM y el modelo de puntos de función para la estimación del tamaño, que ayuda a determinar un conjunto de proyectos homogéneos mediante el uso de una técnica derivada de la estimación por analogía.

Saraç y Duru (2013) presentan una propuesta que combina COCOMO con redes neuronales artificiales incorporando K-means para definir un rango de confianza, estableciendo los límites inferior y superior de la estimación.

1.5 Casos de estudio

Un estudio en Noruega proporcionó una visión general de los métodos de estimación que aplican las compañías en dicho entorno para estimar sus proyectos de software (Moloekken-OEstvold *et al.*, 2004). El método de estimación dominante en estas empresas es el Juicio de Expertos, igual que 10 o 20 años atrás, sin presentar mejoría en los resultados de la estimación al usar métodos formales.

Un estudio acerca de las fortalezas y debilidades de las prácticas de estimación en la industria de software de Pakistán, donde el foco principal es el análisis de las deficiencias de las organizaciones respecto de CMMI Nivel 3, reveló que las compañías utilizan principalmente métodos heurísticos para estimar sus proyectos de software, en especial el Juicio de Expertos en combinación con Delphi, y allí se propone implementar los métodos formales junto a los heurísticos (Nasir y Ahmad, 2006).

Un estudio en la industria de software del estado de Mérida-Venezuela proporciona una visión general acerca del uso de las técnicas de estimación de esfuerzo en dicha industria. Se halla inconformidad en las empresas con el resultado de sus estimaciones porque, al comparar el resultado de la estimación con el valor real del proyecto, hay una tendencia a subestimar el valor real y los métodos formales parecen no estimar apropiadamente el esfuerzo en los proyectos aplicados. En este entorno, las empresas prefieren estimar con la técnica de estimación de puntos de función y la experiencia propia guardando estadísticos de los proyectos que estiman (Villareal, 2007).

Otro estudio en la industria de software de China, basado en la evaluación comparativa de datos de proyectos reales de desarrollo de software y encuestas en estas empresas, muestra una baja utilización de los métodos formales dado el alto costo de adopción y el bajo beneficio de los mismos (Yang *et al.*, 2008).

Un estudio realizado en empresas de software de Kuala Lumpur y Selangor con el fin de conocer las corrientes de estimación de costos de la comunidad de desarrollo de Malasia, nuevamente mostró como preferencia de la industria local del software para estimar sus proyectos el método de juicio de expertos. Sin embargo, el método COCOMO II en esta industria también se utiliza ampliamente. En el estudio se concluye que la integración de estos podría generar estimaciones más precisas de los costos de los proyectos de software (Mansor *et al.*, 2011).

Un estudio realizado en la industria de software de Colombia en ciudades principales como Bogotá, Medellín, Bucaramanga y Cali en empresas certificadas en CMMI-DEV v1.2, permitió encontrar un uso intensivo del método de estimación de juicio de expertos y mostró la necesidad de fortalecer las bases de datos históricas de los proyectos de software o, en su defecto, de usar las bases internacionales de conocimiento para integrar con los modelos de estimación existentes en la industria. Las empresas reconocieron que tienen mucho por mejorar en sus procesos de estimación y se destacó el aporte de las prácticas de calidad como apoyo a la estimación (Páez, 2012).

1.6 Discusión

Las conclusiones que se pueden extraer de los casos de estudios previos se relacionan en:

Tabla 1. Se presenta una relación de los estudios reportados en la literatura, en los cuales se combinan modelos clásicos de estimación con otras técnicas. Se logra identificar que en los últimos años se presenta un interés creciente hacia el desarrollo de métodos de estimación basados en este tipo de combinaciones, entre ellos el juicio de expertos.

Tabla 2. Se presentan evidencias sobre la estimación por juicio de expertos, la cual se emplea ampliamente en las organizaciones de entornos locales como su principal forma de estimación y en la que el experto toma relevancia en la medida que puede considerar aspectos cualitativos que desconocen los modelos existentes, así como las características particulares de la empresa y el entorno en que desarrolla sus actividades.

Tabla1: Combinación de técnicas con nuevos modelos de estimación (Fuente propia).

Nuevos modelos de estimación		(Pérez *et al.* 2006)	(Ayyildiz *et al.* 2006)	(Salvetto De León, P 2006)	(Yahya *et al.* 2008)	(Robiolo 2009)	(Attarzadeh y Hock Ow 2010)	(Mendes, 2012)	(Kaushik *et al.* 2012)	(Torres Ricaurte 2013)	(SARAÇ y DURU 2013)	(Santos 2014)	(Kashyap *et al.* 2014)	(Gharehchopogh *et al.* 2014)	(Maleki *et al.* 2014)	(Almache C. *et al.* 2015)	(Kaur y Singh 2015)
Técnica	Juicio de Expertos	X						X				X					
	Métricas		X			X											
	Estándares de Calidad			X	X					X						X	
	Sof Computing	X	X				X		X		X		X	X	X	X	
Modelo	COCOMO				X				X		X						X
	SLIM																X
	Puntos de Función																X

Tabla 2: Principales conclusiones de los casos de estudio analizados (Fuente propia).

Estudio	Principales conclusiones
(Páez, 2012)	• El Juicio de Expertos es el método de estimación más utilizado. • Se deben desarrollar bases de datos históricas de los proyectos propios de las empresas o en su defecto emplear las bases de conocimiento internacionales adaptadas a los proyectos particulares. • Mejora de procesos: un nivel alto de madurez mejora el desempeño en la estimación de los proyectos de las empresas.
(Mansor *et al.*, 2011)	• El Juicio de Expertos y el modelo algorítmico son los más empleados para estimar. • Integrar métodos produce estimaciones más precisas.

(Yang *et al.*, 2008)	• Es necesaria la mejora de procesos:
	  • Organizacionales: mejorar las tecnologías de estimación, su aceptación y uso e influencia social.
	  • Ingenieriles: mejorar el proceso de desarrollo para hacer frente a la incertidumbre de la estimación de costos.

(Villareal, 2007)	• La experiencia propia y la técnica de Puntos de Función son las más empleadas para estimar. • Los modelos de estimación existentes no estiman adecuadamente el esfuerzo en los proyectos desarrollados por la industria local.
(Nasir y Ahmad, 2006)	• Combinar el enfoque heurístico con modelos formales de estimación produce estimaciones más precisas.
(Moloekken-OEstvold *et al.*, 2004)	• El Juicio de Expertos es el método dominante de estimación. • La mayoría de proyectos de software se subestiman. • La precisión de la estimación no se afecta con el uso de modelos formales. • Los gestores de software creen que la precisión de la estimación de su empresa es mejor de lo que realmente es.

Finalmente, a pesar del auge de las técnicas de *soft computing*, los casos de estudio de empresas de entornos locales muestran que el juicio de expertos en las empresas locales es la técnica más altamente difundida, aceptada y empleada para pronosticar esfuerzos de sus proyectos de desarrollo de software, pero son pocas las nuevas propuestas basadas en la combinación con dicha técnica de estimación. Esta tendencia se refuerza con los conceptos que emiten algunos autores en el tema (Jørgensen, 2004); Jørgensen, Boehm, y Rifkin, 2009); (Macdonell y Shepperd, 2003); (Mendes, 2012); (Robilo, 2013); (Santos, 2014); (Velásquez H., Dyner R., y Souza, 2006), que en sus investigaciones plantean la complementariedad del juicio de expertos con modelos o técnicas para mejorar de manera significativa la precisión de la estimación.

1.7 Objetivos

1.7.1 Objetivo General

Elaborar un caso de estudio sobre los factores que afectan la estimación de proyectos de software con base en el repositorio ISBSG.

1.7.2 Objetivos Específicos

1. Identificar los factores que afectan la estimación de proyectos de software a partir de los modelos relevantes reportados en la literatura.
2. Analizar los factores que afectan la estimación de proyectos de software con base en el repositorio ISBSG.
3. Desarrollar el caso de estudio sobre los factores que afectan la estimación de proyectos de software con base en el repositorio ISBSG.

1.8 Contribuciones

El aporte general de este Trabajo Final de Maestría es el análisis de los factores que afectan la estimación de proyectos de software con base en el caso de estudio repositorio ISBSG.

Las contribuciones específicas son:

1. Una revisión de las técnicas de estimación usadas en pequeñas empresas de entorno local, (ponencia presentada en el VI CIIP -2015 y tercer puesto a mejor trabajo IV jornada RIIPRO Joven 2015 en el VI CIIP -2015).
2. Una Identificación de los factores que afectan la estimación de esfuerzo en proyectos de software pequeños y medianos con base en el repositorio ISBSG.
3. Un análisis de los factores que afectan la estimación de esfuerzo en proyectos de software pequeños y medianos con base en el caso de estudio repositorio ISBSG.
4. Una caracterización de los factores que afectan la estimación de esfuerzo en proyectos de software pequeños y medianos con base en el caso de estudio repositorio ISBSG.

1.9 Organización

Este trabajo se organiza de la siguiente manera: En el Capítulo 1 se presenta el trabajo previo y algunos casos de estudio que aportan a la discusión acerca de la estimación de esfuerzos en el desarrollo del software; en el Capítulo 2 se presenta el marco teórico relacionado con la teoría, modelos y técnicas de estimación de esfuerzo en el software; en el Capítulo 3 se presenta el análisis, identificación y caracterización de los factores que afectan la estimación de proyectos de software con base en el caso de estudio repositorio ISBSG; finalmente, en el Capítulo 4 se presentan las conclusiones y las recomendaciones sugeridas a partir de este trabajo.

2. Estimación de esfuerzo en proyectos de software: fundamentos

2.1 Estimación de esfuerzo

"Estimar consiste en determinar el valor de una variable desconocida a partir de otras conocidas, o de una pequeña cantidad de valores conocidos de esa misma variable. La estimación o predicción es una valoración probabilística donde el valor que se obtiene de una estimación es el centro del rango" (Ruíz Constanten y Cordero Morales, 2013).

La estimación de esfuerzo es el proceso de predecir la cantidad más realista de esfuerzo (expresado en persona/hora) que se requieren para desarrollar o mantener una aplicación o servicio a menudo basado en el conocimiento de aplicaciones o servicios similares previamente desarrollados; se acota con el costo de tiempo y dinero (Aparna *et al.*, 2015; Mendes, 2012).

La estimación de esfuerzo se toma como una de las principales bases de la gestión de proyectos de desarrollo de software, puesto que tiene un papel importante en la creación, desarrollo y mantenimiento de productos. Las estimaciones se usan para determinar la viabilidad de los proyectos, predecir el desarrollo y determinar la cantidad de recursos necesarios. Por ello, es importante la precisión, dado que puede afectar de forma significativa si se entregarán los proyectos a tiempo y dentro del presupuesto (Mendes, 2012).

Las predicciones se pueden utilizar como entrada a los planes, presupuestos y análisis de inversión, a los procesos de fijación de precios y rondas de licitación del proyecto (Aparna *et al.*, 2015). El esfuerzo es un factor determinante del costo total de los proyectos de desarrollo (Mansor *et al.*, 2011; Salvetto De León, P, 2006).

Sin embargo, predecir el esfuerzo en el desarrollo de software es una tarea difícil en la que se deben incluir diversos factores inherentes a la gestión de proyectos, como: incertidumbre, riesgo, planificación, métodos de pronósticos con sus falencias y

limitaciones, políticas organizaciones y socialización de experiencias (Mansor *et al.*, 2011; Mendes, 2012; Velásquez H., Dyner R., y Souza, 2006; Villareal, 2007).

Dado que la estimación de esfuerzo tiene un papel fundamental en la gestión de proyectos de software, la capacidad para obtener una predicción cada vez más certera se convierte en una actividad crítica para las organizaciones, puesto que la precisión de ésta puede determinar el éxito o el fracaso del proyecto (Kaczmarek y Kucharski, 2004; Salvetto De León, P, 2006; Velásquez H. *et al.*, 2006).

La estimación del esfuerzo es importante, entonces, porque permite predecir el costo del desarrollo de software, pero también hacer un continuo seguimiento a los costos y reestimar comparando los costos predichos con los costos reales en cada etapa del desarrollo, lo que ayuda a evidenciar el estado del proyecto y a identificar a tiempo posibles correcciones en el calendario y los costos, mejorando la calidad de la estimación y, en última instancia, del producto software. La estimación del esfuerzo además permite (Rajkumar y Alagarsamy, 2013; Salvetto De León, P, 2006):

- Clasificar y priorizar los proyectos de desarrollo de acuerdo con un plan de negocios general.
- Asignar los recursos que se comprometerán con el proyecto y su uso.
- Evaluar el impacto de los cambios y reprogramación en los proyectos.
- Facilitar la administración y control de los proyectos cuando los recursos se adaptan mejor a las necesidades reales.
- Permitir que los resultados sean más consistentes con lo planificado.

El resultado de la estimación puede hacer la diferencia entre ganancia o pérdida de recursos y confianza en la industria (Mansor *et al.*, 2011). Una estimación errada puede dar dos resultados: una subestimación; generando proyectos que exceden sus presupuestos e incumplen las fechas programadas, comprometiendo la calidad, o una sobreestimación, generando proyectos excesivos en costos y recursos, contribuyendo a la pérdida de oportunidades de negocio (Salvetto De León, P, 2006).

2.2 Factores que impactan la estimación

Los proyectos de desarrollo de software, como cualquier proyecto, deben iniciar con un buen plan, lo que se dificulta dada la complejidad de realizar una adecuada y realista estimación del esfuerzo. El proceso para crear una adecuada planeación consta de: la estimación del tamaño del producto, la estimación del esfuerzo necesario para su desarrollo y la estimación de la duración. Para ello, en general, los modelos de estimación se basan en un conjunto factores significativos de entrada que generan una salida: el esfuerzo (Ruíz Constanten y Cordero Morales, 2013).

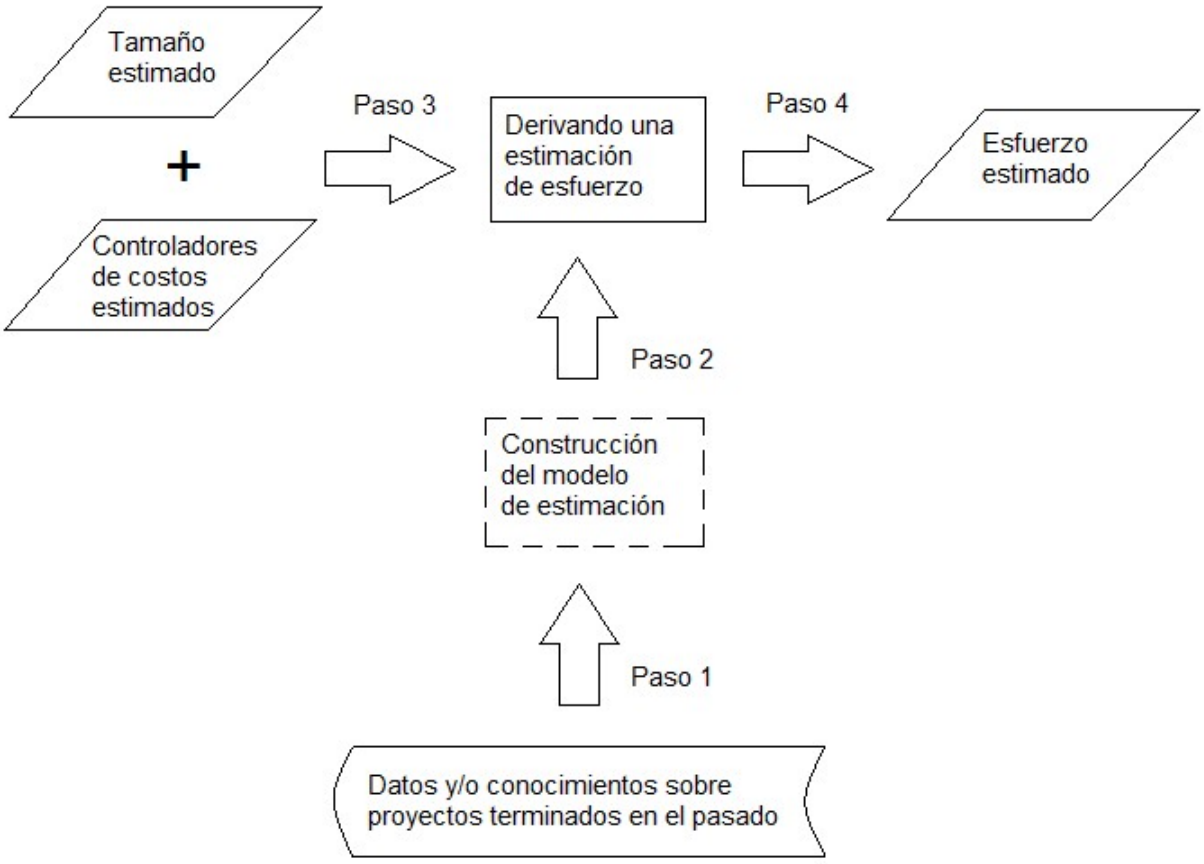
Ruíz Constanten y Cordero Morales (2013) mencionan cuatro factores que influyen significativamente en las estimaciones: la complejidad y el tamaño del proyecto, el grado de incertidumbre estructural y la disponibilidad de información histórica. Estos factores dan cuenta de la experiencia de la organización tanto en el proceso propio del desarrollo de software como en la gestión de proyectos de software y su estimación.

Al no disponer de todos los medios y de la información necesaria, las estimaciones de los proyectos de software suelen errar, normalmente, pronosticando resultados subestimados o sobreestimados al real. Por ello, no incluir y/o no conocer y valorar los factores que inciden en una estimación como causa de la mala planeación y/o ejecución de un proyecto causa pérdidas esencialmente de dinero y tiempo (Ruíz Constanten y Cordero Morales, 2013).

2.3 Pasos para obtener una estimación de esfuerzo

La premisa del proceso es que la estimación de esfuerzo de desarrollo de software es un proceso basado en experiencias previas. Algunos o todos los pasos se pueden ejecutar más de una vez a lo largo de un ciclo de desarrollo de software determinado, dependiendo del modelo de proceso del proyecto adoptado. Este define en qué etapa se obtiene una estimación de esfuerzo y si esta estimación se vuelve a revisar, modificar y actualizar en algún momento a lo largo del ciclo de vida de desarrollo de un proyecto (Mendes, 2012), como se muestra en la **Figura 1:**

Figura 1: Pasos para obtener una estimación de esfuerzo (Mendes, 2008).



2.4 Cálculo de la estimación de esfuerzo

Basado en una distribución beta, la estimación de esfuerzo E se calcula, como: $\boldsymbol{E} = (\boldsymbol{o} + \boldsymbol{4r} + \boldsymbol{p})/\boldsymbol{6}$, donde $\boldsymbol{o}$ es una estimación optimista, $\boldsymbol{r}$ es una estimación realista y $\boldsymbol{p}$ es una estimación pesimista (Mendes, 2008).

2.5 Medición de la precisión de la predicción de esfuerzo

Las estadísticas más usadas para medir la precisión de la predicción de esfuerzo son: (Mendes, 2008):

- La magnitud del error relativo, definida como: $\boldsymbol{MRE} = \frac{|\boldsymbol{e}-\hat{\boldsymbol{e}}|}{\boldsymbol{e}}$ donde $\boldsymbol{e}$ es el esfuerzo real de un proyecto en el conjunto de validación y $\boldsymbol{E}$ es el esfuerzo estimado que se obtuvo utilizando el modelo $\boldsymbol{m}$ o técnica $\boldsymbol{t}$.
- El valor $\boldsymbol{L}$ generalmente fijado en el 25% o (0.25). Medida que evalúa si el error relativo asociado con un proyecto no es mayor que $\boldsymbol{L}$.
- El Residuo absoluto, es la diferencia absoluta entre el esfuerzo real y el estimado.

Modelos estadísticos para medir la precisión de la predicción de esfuerzo son (Mendes, 2008) :

- La magnitud media de error relativo (MMRE)
- La magnitud media de error relativo (MdMRE)
- La predicción en el nivel n (Pred [n])

2.6 Modelos de estimación

“Los modelos de estimación de esfuerzo, son considerados métodos o metodologías de análisis y evaluación de los factores que intervienen en un determinado desarrollo” (Ramírez, 2011). Existen numerosos métodos de estimación, pero frecuentemente no son capaces de generar las estimaciones correctas y necesarias para la gestión y planificación de proyectos de desarrollo. Generalmente, los proyectos fracasan porque no terminan bajo el tiempo y el presupuesto asignado. Muchos factores influyen en la falla, como la inconsistencia en los requisitos de software, las dificultades técnicas, la falta de experiencia y conocimiento al estimar, factores humanos, organizaciones y otros (Kaczmarek y Kucharski, 2004; Pushkar y Kumari, 2013; Shepperd y MacDonell, 2012).

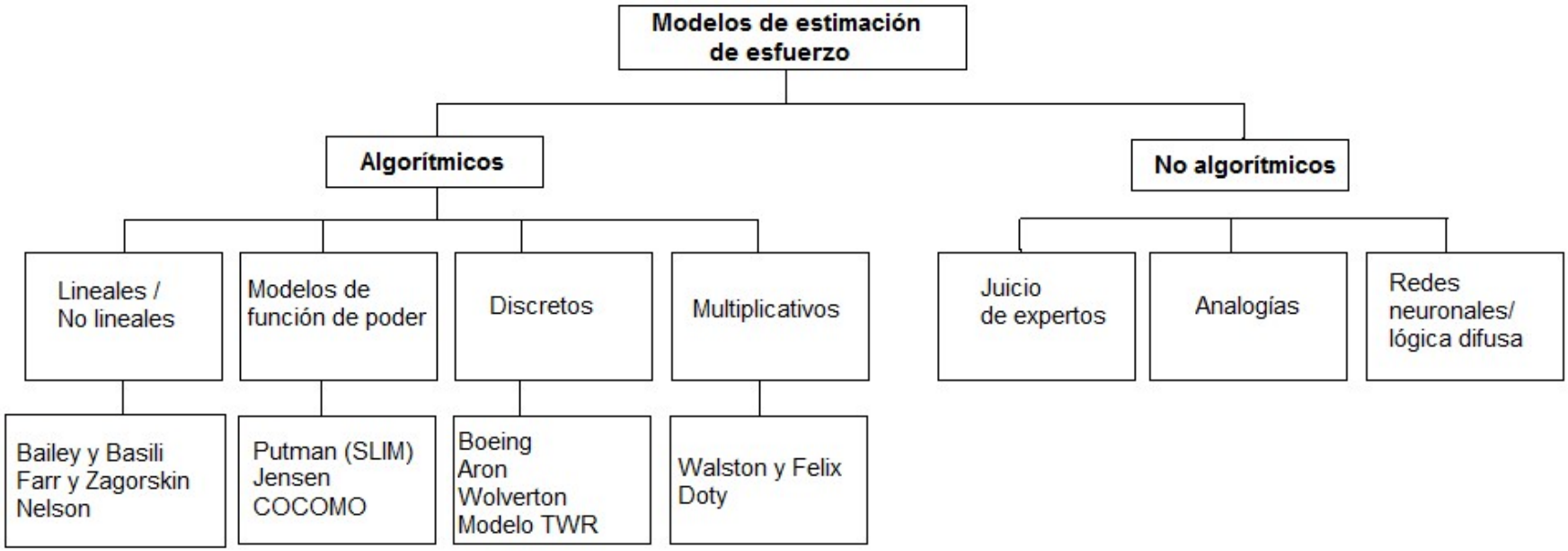
Los primero modelos matemáticos propuestos fueron los paramétricos, que se fundamentan en ecuaciones matemáticas mediante las cuales, a partir de un conjunto de variables independientes de entrada como el tamaño del proyecto, se obtiene el valor de un conjunto de variables dependientes de salida como el esfuerzo. Generalmente, el proceso consta de dos pasos; primero, se genera una primera estimación a partir en función de un conjunto reducido de variables independiente y segundo, se precisa el resultado mediante otro conjunto de variables que permiten refinar los cálculos (Ruíz Constanten y Cordero Morales, 2013).

A partir de estos modelos, desde los años 60 y hasta la actualidad, con el ánimo de mejorar la exactitud de la estimación de costos desarrollo y así lograr asignar de forma adecuada los presupuestos, surgen varias técnicas de estimación de esfuerzo, que se pueden dividir en tres (3) amplias categorías: modelos algorítmicos, estimación basada en el juicio de expertos y técnicas de inteligencia artificial (Mendes, 2012).

2.7 Categorización de los modelos de estimación de esfuerzo

Existen diversas categorizaciones de los modelos de estimación de esfuerzo, la mayoría divididos en dos categorías principales: modelos algorítmicos y modelos No algorítmicos. Una categorización se muestra en la **Figura 2:**

Figura 2: Categorización de modelos de estimación de esfuerzo (Aparna *et al.*, 2015).



2.7.1 Modelos Algorítmicos:

Basados en algoritmos especiales que requieren datos de entrada, como por ejemplo, insumos de investigación, datos históricos y de uso, tales como SLOC (*Source lines of code*), número de funciones a realizar y otros factores de costo tales como el lenguaje, la metodología de diseño, los niveles de habilidad, evaluaciones de riesgo, entre otros; generan resultados mediante el uso de relaciones matemáticas proporcionando ecuaciones para llevar a cabo la estimación de software. Tienen la forma general: $VDE = f(fc_{1,} fc_{2,} fc_3, \dots, fc_k)$ donde VDE es la variable independiente a estimar y fc_i son los factores de costo. Los modelos algorítmicos se dividen en dos tipos principales: ***estáticos***, permiten pronosticar el valor de las variables dependientes en función de un conjunto de factores de costo independientes y no incluyen en su ecuación el tiempo; d***inámicos***, variables interdependientes, incluyen el tiempo dentro de las ecuaciones del modelo, reconociendo la evolución de los proyectos a lo largo del tiempo en donde el esfuerzo y los factores de costo no permanecen estáticos. Algunos de los algoritmos son: los modelos lineales, los modelos exponenciales, los modelos discretos y los modelos compuestos. La desventaja de los modelos algorítmicos radica en la incapacidad para producir estimaciones en situaciones especiales y con pocos datos de entrada, arrojando estimaciones inexactas en dichos casos. (Aparna ***et al.***, 2015; Khatibi y Jawawi, 2010; Rajkumar y Alagarsamy, 2013; Salvetto De León, P, 2006).

- **Modelos lineales:** son de la forma $E = a_0 + a_i x_i + \cdots + a_n x_n$ donde el esfuerzo (E), corresponde a la conjunción de $x_{i,} \dots, x_n$ que son los factores de costo y $a_{0,} \dots, a_n$ que son un conjunto de coeficientes elegidos para ajustarse lo mejor posible a los datos de proyectos concluidos. El costo de un desarrollo se obtiene multiplicando el esfuerzo (E) por una “constante” de trabajo. Empleados en modelos como Farr y Zagorski, Nelson, Bailey y Basili, entre otros (Aparna ***et al.***, 2015; Ramírez, 2011).

- **Modelos multiplicativos o exponenciales:** son de la forma $E = a0_{j^k}, \dots, a_{n^k} x_n$ donde el conjunto x_i son los factores de costo y el conjunto a_i son los coeficientes elegidos para ajustarse lo mejor posible a los datos de proyectos concluidos. Las variables de costo se consideran restringidas de manera discreta, solo pueden tomar valores -1, 0 y 1. Empleados en modelos como Walston Felix y Doty, entre otros (Aparna ***et al.***, 2015; Ramírez, 2011).

- **Modelos discretos:** tienen una forma tabular y permiten relacionar factores de costo como esfuerzo, tiempo y dificultad. La información de las variables de costo se dispone en tablas de datos que relacionan estos valores con el esfuerzo en las distintas fases del desarrollo de software. En ocasiones, se relacionan también con multiplicadores utilizados para ajustar el esfuerzo estimado. Pueden llegar a tener muchas tablas, demasiados datos para calibrar y requerir bases de datos para contrastar y validar dichos valores. Empleados en modelos como Aaron Sloman, Wolverton y Boeing, entre otros (Aparna *et al.*, 2015; Ramírez, 2011; Salvetto De León, P, 2006).

- **Modelos compuestos:** incorporan características de los modelos lineales, exponenciales y discretos para estimar el esfuerzo del desarrollo y el análisis de los factores de costo. Se da combinando dos o más enfoques. Utilizan la forma funcional más apropiada para cada componente implicado en la estimación del costo del software. Requieren contrastar y validar grandes cantidades de datos, son de complejo aprendizaje y uso. Empleados en modelos como RCA PRICES, SLIM, TRW SCEP, COCOMO en sus diferentes versiones y FFP, entre otros (Ramírez, 2011).

2.7.2 Modelos No Algorítmicos:

"Estos modelos usan el conocimiento de expertos familiarizados con un dominio. Las estimaciones obtenidas están basadas en la síntesis de los resultados obtenidos en los proyectos en los que participaron, procurando extraer conocimientos tácitos y no formalizados del que dispone el experto" (Salvetto De León, P, 2006). Se pueden obtener estimaciones rápidamente y se adaptan a situaciones especiales. Son muy útiles cuando no se dispone de información empírica cuantitativa o no hay precedentes. Su desventaja se encuentra la subjetividad y la dependencia a las personas expertas. (Khatibi y Jawawi, 2010; Salvetto De León, P, 2006; Santos, 2014; Tailor *et al.*, 2014). Se dividen en tres tipos principales*: juicio de expertos, estimación por analogías y redes neuronales artificiales (RNA).*

Juicio de expertos: "Juicio de expertos es cuando la estimación se basa en la experiencia de una o más personas que están familiarizadas con el desarrollo de aplicaciones de software similares a la que se intenta estimar" (Hughes, 1996). La estimación experta se

basa en un proceso de razonamiento no explícito ni replicable, es decir la intuición. Incluso, los métodos formales de estimación de esfuerzo requieren del juicio experto como parámetro de entrada (Santos, 2014). Basados en una distribución beta la estimación del esfuerzo E se calcula, como: $E = (o + 4r + p)/6$ donde o es una estimación optimista, r es una estimación realista y p es una estimación pesimista (Mendes, 2008).

Estimación por analogías: consiste en razonar por analogía o simple comparación con uno o más proyectos completados, cuál de ellos o cuál caso se asemeja más al proyecto que se pretende estimar (Ramírez, 2011). El principal supuesto es que proyectos similares tendrán costos y tiempos similares. Puede emplearse a nivel de proyecto o para estimar determinados subsistemas. Para ello, se analizan detalladamente las decisiones técnicas y gerenciales tomadas en los proyectos estudiados y luego se extrapolan los resultados al proyecto que se está estimando. El poder estimar con base en experiencias o proyectos reales se entiende como una ventaja; la desventaja se encuentra en determinar hasta qué punto es aplicable la experiencia anterior al nuevo proyecto (Ramírez, 2011; Salvetto De León, P, 2006).

Redes neuronales artificiales (RNA): inspiradas en las redes neuronales biológicas, donde procesadores interconectados entre sí para producir una salida representan las neuronas, las cuales, tienen la capacidad de hacer una suma ponderada de los estímulos recibidos. Si la suma supera un umbral, se genera una salida que puede ir hacia otras neuronas o formar parte de la salida. Existen dos tipos principales de redes neuronales: *feed-forward*, no tienen *look* y la alimentación se da hacia delante, son estáticas y la respuesta siempre es la misma ante el mismo estímulo, por tanto no guardan memoria de estados anteriores; y *feedback* que tienen *look* recursivos. Las redes se entrenan usando datos históricos para mejorar los resultados reduciendo la diferencia entre los datos reales y sus predicciones dentro de un nivel de tolerancia prefijado. Son eficientes para resolver problemas complejos, tienen inconvenientes estadísticos con los datos de entrenamiento, requieren un volumen alto de datos raramente disponible y los resultados son poco intuitivos (Salvetto De León, P, 2006).

2.8 Ventajas y Desventajas de los modelos de estimación de esfuerzo

En la literatura se encuentra un amplio número de modelos tanto algorítmicos como no algorítmicos. Estos modelos poseen ciertas ventajas y desventajas frente a la necesidad específica del proyecto de software, como pueden ser el tamaño o las situaciones especiales del mismo. En la **Tabla 3**, se presenta un resumen de las ventajas y desventajas generales de los modelos de estimación en sus dos grandes clasificaciones: algorítmicos y no algorítmicos. En la **Tabla 4**, se presenta un resumen de las ventajas y desventajas específicas de algunos modelos de estimación tanto algorítmicos como no algorítmicos.

Tabla 3: Ventajas y desventajas de los modelos algorítmicos y no algorítmicos (Aparna *et al.*, 2015).

Modelo	Ventajas	Desventajas
Modelo algorítmico	• Permite generar estimaciones repetibles. • Es eficiente porque puede mantener una familia de estimaciones o un análisis de sensibilidad. • Facilita la modificación de los datos de entrada, el perfeccionamiento personalización de fórmulas. • Se calibra de manera objetiva de acuerdo con la experiencia anterior.	• Es incapaz de producir estimaciones en situaciones especiales y con pocos datos de entrada, arrojando estimaciones inexactas. • Requiere mucha información de entrada; de lo contrario, genera estimaciones inexactas. • La experiencia y algunos factores no se pueden cuantificar fácilmente.
Modelo no algorítmico	• Tiene una amplia credibilidad empresarial porque el experto puede incorporar experiencias pasadas que no están en la información histórica y que son difíciles de incorporar en los modelos matemáticos, como Información contextual y cualitativa de posibles eventos futuros. • Permite tener en cuenta las experiencias y requisitos de proyectos pasados como base para nuevos proyectos. • Permite tener en cuenta factores de costos como las características y las interacciones personales y organizacionales, que son difíciles de cuantificar. • Los pronósticos se realizan en menos tiempo y con mayor facilidad.	• Un factor que afecta la exactitud en las predicciones de esfuerzo es la incertidumbre en el tamaño del software, normalmente medido en líneas de código. • A medida que aumentan los factores a considerar y que las relaciones entre estos se hace más complejas, el pronóstico se degrada. • Como actividad humana se afecta con las propias limitaciones de la mente humana y el modelo mental adquirido mediante la experiencia personal del experto y que representa el conocimiento subjetivo que este posee.

Tabla 4: Ventajas y Desventajas de algunos modelos de estimación (Fuente propia).

Método	Clasificación	Ventajas	Desventajas	Referencias
SLOC	**Algorítmico**	• Fácil de implementar para estimar el tamaño del software.	• Difícil de implementar en las primeras etapas de desarrollo, poco efectivo en proyectos grandes, depende del lenguaje de programación.	(Tailor *et al*., 2014).
FPA	**Algorítmico**	• Cualquier lenguaje de programación, genera mejores resultados que SLOC.	• Difícil de implementar, no considera la calidad de la producción.	(Khatibi y Jawawi, 2010). (Pushkar y Kumari, 2013).
SEER-SEM	**Algorítmico**	• Efectivo en proyectos muy grandes.	• Requiere muchos parámetros de entrada aumentado la complejidad y la incertidumbre.	(Tailor *et al*., 2014).
COCOMO	**Algorítmico**	• Bueno para generar estimaciones tempranas de costos. • En su versión actualizada, proporciona más apoyo a los procesos de desarrollo mediante la actualización de las bases de datos de proyectos, la reutilización de código y el procesamiento por lotes. • Genera resultados claros y es efectivo en proyectos pequeños.	• Precisión limitada en proyectos grandes, requiere muchos parámetros que aumentan la complejidad y el tiempo de estimación. No se adapta a cualquier tipo de proyecto.	(Khatibi y Jawawi, 2010). (Pushkar y Kumari, 2013).
Modelo Lineal	**Algorítmico**	• Es el mejor método de predicción utilizando la técnica de regresión lineal.	• La precisión de la estimación no es confiable, presenta diferencias entre el error real y el estimado.	(Tailor *et al*., 2014).
SLIM	**Algorítmico**	• Modelo probabilístico empleado en proyectos grandes.	• No es funcional en proyectos pequeños.	(Tailor *et al*., 2014).
Juicio de Expertos	**No Algorítmico**	• Se puede estimar con rapidez, se adapta a proyectos especiales. • Los expertos procesan la información (la existencia o falta de ésta) con mayor flexibilidad.	• La precisión de la estimación depende de la experiencia del experto. • La estimación es subjetiva depende de los sentimientos y lógica del experto. • El grado de inconsistencias y la ponderación de variables.	(Khatibi y Jawawi, 2010). (Jørgensen, 2004). (Pushkar y Kumari, 2013). (Robiolo y Santos, 2013). (Tailor *et al*., 2014).
RNA	**No Algorítmico**	• Se pueden desarrollar modelos precisos por medio de un ejemplo dado.	• Fallan cuando las condiciones cambian y los modelos no se puede recalibrar, por lo que cada caso nuevo genera un nuevo modelo.	(Ferreira *et al*., 2014).
Lógica Difusa	**No Algorítmico**	• No requiere entrenamiento previo, flexible.	• Difícil de utilizar.	(Khatibi y Jawawi, 2010). (Tailor *et al*., 2014).

2.9 Estimación por Juicio de Expertos

Estimación por “Juicio de expertos es cuando la estimación se basa en la experiencia de una o más personas familiarizadas con el desarrollo de aplicaciones de software similares a la que se intenta estimar” (Hughes, 1996). La estimación experta se basa en un proceso de razonamiento no explícito ni replicable, es decir la intuición. Incluso, los métodos formales de estimación de esfuerzo requieren juicio experto como parámetro de entrada (Santos, 2014).

Según Jorgensen y Shepperd (2007), Jørgensen (2004), Jørgensen (2007), Jørgensen, Boehm y Rifkin (2009), el método más aceptado en la industria de software para la estimación de esfuerzo es el juicio de expertos.

Jørgensen *et al.* (2009) argumentan que el juicio de expertos ofrece mayor precisión que los métodos formales de estimación porque el experto en el dominio del desarrollo de software tiene mayor información del contexto que los modelos y en situaciones inestables sus predicciones son mejores.

2.10 Juicio de expertos en combinación con modelos y técnicas

Aunque son pocas las propuestas, varios autores encuentran beneficios en la combinación del juicio de expertos con los modelos y técnicas de estimación. Jørgensen (2004) y Jørgensen *et al.* (2009) reconocen la importancia del juicio de expertos pero, además, sugieren la necesidad de combinar esta técnica con modelos en la búsqueda de una mejor estimación.

Macdonell y Shepperd (2003) realizan la misma sugerencia dada el alto grado de interdependencia entre las estimaciones basadas en los modelos clásicos de estimación y la estimación por expertos, lo que dificulta seleccionar el método de estimación más preciso. Por ello, proponen como solución la combinación de los modelos (Robiolo y Santos, 2013).

Velásquez H. *et al.* (2006) también encuentran fortalezas en dicha combinación como técnicas complementarias que pueden mejorar de manera significativa la precisión de la estimación. Cada técnica puede suplir falencias de la otra. El juicio de expertos tiene la fortaleza de permitir la inclusión mental de elementos contextuales y cualitativos que usualmente son un problema mayor en los modelos, pero tiene relacionadas falencias propias de la mente humana, que pueden llegar a simplificar demasiado el modelo para generar pronósticos válidos. Los modelos matemáticos, por su parte, presentan soluciones para algunos de sus problemas fundamentales pero carecen de la capacidad para manejar información.

Según Velásquez H. *et al.*, (2006)*, e*n términos de esta complementariedad de técnicas, se dan posibilidades como que el juicio experto puede ser una variable numérica de entrada al modelo matemático o que se puede ajustar con las predicciones estadísticas, lo que se denomina *intervención del pronóstico* con las siguientes ventajas:

- La integración dentro del modelo de: información contextual, cualitativa de difícil medición o cuantitativa que no se puede incorporar en el modelo, de otras variables que no existían al construir el modelo y/o cambios estructurales que ocurren y para los cuales no se cuenta con información histórica para calibrar nuevamente el modelo.
- Existe retroalimentación entre el juicio experto y el pronóstico del modelo.

Velásquez H. *et al.* (2006) también indican que para mantener la calidad de las predicciones al realizar ajustes de pronósticos mediante la combinación de estas técnicas, se deben respetar ciertas reglas:

- Registrar las razones por las que se realiza el ajuste del pronóstico, para luego evaluar su desempeño y evaluar el impacto en el pronóstico.
- Los ajustes se deben realizar bajo escenarios específicos, relacionados con la ocurrencia de hechos futuros que afecten el desempeño del modelo matemático.
- Definir en qué condiciones no se debe ajustar el pronóstico.

Por su parte Mendes (2012) propone un enfoque centrado en el experto en combinación con una técnica que permita incluir la incertidumbre y la casualidad, como las redes bayesianas. En este mismo enfoque en el experto (Robiolo y Santos, 2013; Santos, 2014) se propone la combinación con estándares de calidad.

2.11 Las doce (12) buenas prácticas para la estimación por juicio de expertos

Jørgensen (2004) ofrece una lista de doce (12) buenas prácticas, validadas empíricamente, para la estimación por juicio de expertos que pueden facilitar la labor del estimador y mejorar sus pronósticos (Santos, 2014):

1. Evaluar la precisión de la estimación pero evitar la alta presión de evaluación. Al evaluar la precisión hay que evitar la presión sobre el estimador que se da especialmente en términos de dinero y tiempo.
2. Evitar las metas de estimación en conflicto. Esto se puede evidenciar cuando hay intereses particulares de personas como los *stakeholders*, diferentes a la meta de precisión en sí, en donde la estimación puede ser sesgada y el experto puede producir estimaciones poco realistas.
3. Pedir a los estimadores justificar y criticar sus estimaciones. La autocrítica permite mejorar y retroalimentar el proceso de la estimación.
4. Evitar información irrelevante y poco fiable. Este tipo de información afecta de manera negativa la estimación del experto.
5. Utilizar los datos documentados de las tareas de desarrollo anteriores. Se evita la dependencia a la memoria y, por tanto, el error humano, mejorando la precisión de la estimación.
6. Encontrar expertos en el dominio y un buen registro o antecedentes de estimaciones.
7. Estimar utilizando las técnicas *Top-Down* y *Bottom-up* de manera independiente. Utilizarla de manera independiente evita la influencia del otro.
8. Usar listas de comprobación (*checklist)*. Estas evitan olvidar actividades y subestimar el esfuerzo requerido especialmente en eventos inesperados.
9. Combinar las estimaciones de diferentes expertos y estrategias de estimación. La complementariedad permite mejorar la precisión de la estimación.
10. Evaluar la incertidumbre de la estimación.

11. Proporcionar información sobre la precisión de la estimación y las relaciones entre tareas.
12. Proporcionar oportunidades de formación en estimación.

2.12 Algunas técnicas de estimación basados en juicio de expertos

En la estimación por juicio de expertos, un desarrollador o gestor describe los parámetros del proyecto y los expertos hacen estimaciones basadas en su experiencia. Se puede sistematizar y mejorar la opinión de los expertos mediante las siguientes técnicas:

2.12.1 Delphi:

Es una de las técnicas de estimación por juicio de expertos más usadas. Desarrollada en los años cuarenta en Estados Unidos por la Corporación RAND a partir de la necesidad de unificar opiniones de expertos en un proyecto de las Fuerzas Armadas, se comienza a utilizar en el desarrollo de software a partir de 1981 (Boehm, 1981). Esta técnica buscar unificar las opiniones de los expertos y obtener consenso por medio de la recopilación de las opiniones de los diferentes expertos mediante cuestionarios anónimos y con retroalimentación controlada con el objetivo de producir una estimación sin sesgo y precisa (Del Valle Roque *et al.*, 2014; Pushkar y Kumari, 2013; Rajkumar y Alagarsamy, 2013; Santos, 2014; Suri y Ranjan, 2012; Tailor *et al.*, 2014). Esta técnica estructurada de la opinión de los expertos consta de los siguientes pasos:

1. Selección de los expertos.
2. Información para los Expertos
3. Clasificación de las estimaciones de los expertos
4. La convergencia de las estimaciones y finalización

2.12.2 Wideba Delphi:

Técnica derivada de la técnica Delphi, que se basa en lograr el conceso en la estimación de un grupo de expertos (Boehm, 1981). Incluye mucha más interacción y comunicación entre los participantes, a diferencia de la técnica Delphi original donde se evitan las discusiones de grupo. Esta técnica consta de los siguientes pasos (Santos, 2014):

1. El coordinador le presenta a cada experto la especificación de la actividad/tarea y un formulario de estimación.
2. Los estimadores preparan sus estimaciones individualmente.
3. El coordinador llama a reunión donde los expertos discuten los problemas de estimación relacionados con el proyecto.
4. Los expertos llenan los formularios anónimamente y se los entregan al coordinador. El formulario tiene un rango de 1 a 20 meses e inicialmente es tres veces mayor que el rango que se espera que los estimadores usen para que no se sientan restringidos a un rango predefinido.
5. El coordinador prepara un resumen de las estimaciones en un formulario por iteración y lo distribuye entre los estimadores para que vean como sus estimaciones difieren de las estimaciones de los otros estimadores.
6. El coordinador llama a reunión donde se focalizará en tener a los expertos discutiendo los puntos donde las estimaciones variaron ampliamente.
7. Los expertos votan anónimamente si aceptan el promedio de estimación. Si algunos de los estimadores vota que no, entonces se vuelve al paso 3.
8. La estimación final es un solo valor que se obtiene del ejercicio de *Delphi*.

2.12.3 Analogías:

Es un enfoque más formal que el juicio de expertos. En esta técnica, los expertos comparan el proyecto propuesto con uno o más proyectos anteriores intentando encontrar similitudes y diferencias particulares. Para ello, debe existir una base de proyectos terminados que se usa para buscar aquellos que más se asemejen al proyecto que se quiere estimar. La técnica se basa en caracterizar proyectos en términos de características similares como: el número de interfaces, el método de desarrollo o el tamaño de los requisitos funcionales. Los pasos de esta técnica son (Santos, 2014; Tailor *et al.*, 2014):

1. Elegir la analogía.
2. Investigar similitudes y diferencias.
3. Examinar la calidad de analogía.
4. Proporcionar la estimación.

La estimación por analogías es una estrategia de los sistemas CBR (Razonamiento Basado en Casos), que a su vez forman parte de las máquinas de aprendizaje junto con Redes Neuronales y Reglas de Inducción. Los pasos de ésta técnica son (Santos, 2014):

1. Caracterización de casos.
2. Almacenamiento de casos pasados.
3. Recuperación de casos similares para usar analogías.
4. Utilización del caso recuperado para resolver el caso del problema en cuestión conocido como caso de adaptación.

2.12.4 Top Down:

En esta técnica se usa información histórica de proyectos anteriores y similares como punto de partida para las estimaciones de esfuerzo. Se usa en la fase inicial de planificación de los proyectos, cuando no se conoce mucha información sobre los requisitos de los mismos y se puede utilizar WBS (*Work Breakdown Structure*) a alto nivel. En este enfoque, la estimación de los costos totales del sistema se deriva de las propiedades globales, ya sea utilizando métodos algorítmicos o no algorítmicos. El costo total se puede dividir entre los diversos componentes (Rajkumar y Alagarsamy, 2013; Santos, 2014).

2.12.5 Bottom Up:

También llamada descomposición aditiva. Aplica el enfoque WBS a bajo nivel, se descompone el proyecto en pequeños paquetes. En esta técnica se estima el costo de cada uno de los componentes del software y luego se combinan los resultados para llegar a un costo estimado del proyecto en general. La estimación se produce a partir de los conocimientos acumulados con los pequeños componentes de software y sus interacciones (Pushkar y Kumari, 2013; Santos, 2014).

2.12.6 Price to Win:

La estimación se basa en el presupuesto del cliente en lugar, de la funcionalidad del software. Se le informa al cliente sobre los gastos estimados y éste decide si los puede asumir o no. Si no es así, el estimador modifica la estimación para encajar el esfuerzo y se renegocia con el cliente, generalmente reduciendo los requisitos del sistema (Pushkar y Kumari, 2013).

2.12.7 Planning Poker:

Es una variación del método *Wideba Delphi*, técnica para llegar a una estimación basada en el consenso del equipo de expertos, utilizada mayormente en el desarrollo ágil de software, en particular en *Extreme Programming*. El equipo de estimación lo conforman todos los desarrolladores, generalmente no más de 10 personas: probadores, analistas, diseñadores, desarrolladores, etc. Es un enfoque que combina opiniones de expertos, analogías y desagregación. Se basa en una lista de características, generalmente historias de usuario, que describen el software a desarrollar y un mazo de cartas con estimaciones validadas. Generalmente el mazo de cartas muestra la secuencia Fibonacci incluyendo un cero: 0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, para reflejar la incertidumbre inherente en la estimación, u otras progresiones similares: 0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 8 y dos tarjetas adicionales una con el signo de interrogación (¿) y otra con el signo de infinito (∞), que se usan para declarar completa incertidumbre o desconocimiento de la característica (Santos, 2014).

2.13 Ventajas y desventajas de las técnicas de estimación por juicio de expertos

El juicio de expertos, como todo modelo, presenta ciertas ventajas y desventajas frente a las particularidades de los proyectos, en la **Tabla 5** se presenta un resumen general.

Tabla 5: Ventajas y desventajas de las técnicas de estimación por juicio de expertos (Fuente propia).

Método	Ventajas	Desventajas	Referencias
Delphi / Wideba Delphi	Libre de la presión social, la influencia de la personalidad y el dominio individual. Permite el intercambio de información y discusión entre los participantes. Ofrece una amplia perspectiva de análisis sobre los problemas y cuestiones. Facilita la obtención de consensos entre los grupos.	Las sentencias son las de un grupo seleccionado y pueden no representar la opinión prevaleciente. Requiere habilidad en la comunicación escrita. Requiere tiempo adecuado y el compromiso de los participantes.	(Rajkumar y Alagarsamy, 2013).
Analogías	La estimación se basa en la experiencia de proyectos reales. Permite identificar las diferencias entre los proyectos completados y los propuestos y estimar los impactos.	Se requiere gran información de proyectos similares anteriores con la que no siempre se cuenta. Se requiere calibrar el modelo.	(Pushkar y Kumari, 2013). (Rajkumar y Alagarsamy, 2013).
Top Down	Se centra en las actividades a nivel de sistema, tales como la integración, documentación, gestión de configuración, etc., que se pueden ignorar en otros métodos de estimación, por lo que no se pierde el costo de las funciones de dicho nivel. Requiere un mínimo de detalle del proyecto. Es más rápido fácil de implementar.	No identifica los problemas difíciles de bajo nivel que son propensos a aumentar los costos. No proporciona una base detallada para justificar las decisiones o estimaciones.	(Pushkar y Kumari, 2013). (Rajkumar y Alagarsamy, 2013).
Bottom Up	Detallado y estable. Fomenta el compromiso individual.	Puede pasar por alto los costos a nivel de sistema. Requiere más esfuerzo, tiempo.	(Pushkar y Kumari, 2013). (Rajkumar y Alagarsamy, 2013).
Price to win	Tiene en cuanta el presupuesto del cliente por lo que facilita el proceso y obtención de la licitación.	Valora el presupuesto sobre la funcionalidad. Esto no es una buena práctica, porque puede causar retrasos graves en la entrega u obligar al equipo de desarrollo a trabajar horas extras.	(Pushkar y Kumari, 2013).
Planning Poker	Útil a la hora de discutir estrategias de implementación de cada tarea. Otorga una mejor visión del trabajo de cada desarrollador. Reduce el problema de “anclaje” al revelar simultáneamente todas las estimaciones del grupo. Mejora la eficiencia en las estimaciones sobre grupos desestructurados.	Requiere la experiencia del equipo en tareas similares. Incrementa los valores extremos.	(Santos, 2014).

3. Caso de estudio repositorio ISBSG

ISBSG (*International Software Benchmarking Standards Group*) es una organización sin ánimo de lucro, fundada en 1997 por un grupo de asociaciones nacionales de métricas de software. Los actuales socios son: *China System and Software Process Improvement* (SPI) *Association*, *Beijing Kexin Science and Technology Ltd*, GUFPI-ISMA (*Gruppo Utenti Function Point Italia – Italian Software Metrics Association*), JFPUG (*Japan Function Point User Group*), AMMS (*Mexican Association of Software Metrics*), NESMA (*Nederlandse Software Metrieken Gebruikers*), *Metrics Quest*, LEDA MC, Swiss ICT, Galorath, IFPUG (*International Function Point Users Group*), COSMIC (*Common Software Measurement International Consortium)* y Dinh Bao Tuyen (ISBSG, 2016a).

Su objetivo es promover el uso de datos de la industria de TI para mejorar los procesos y productos de software. Para ello, cuenta con dos repositorios de datos: de mantenimiento y soporte y de desarrollo y mejora de software, que sirven como punto de referencia y comparación para nuevos proyecto de TI a nivel mundial, en temas como: estimación, *benchmarking*, gestión de proyectos, planificación de infraestructura, planificación de ofertas, gestión de subcontratación (*outsourcing)*, cumplimiento de estándares y apoyo presupuestario (ISBSG, 2016a).

Todos los datos que recibe la ISBSG se verifican antes de agregarlos a los repositorios. A cada proyecto se le asigna un código de clasificación (A, B, C o D) para denotar su calificación de calidad (ISBSG, 2016a), así:

A = Los datos enviados se evaluaron como sólidos sin identificar nada que pudiera afectar su integridad.
B = La presentación parece fundamentalmente sólida, pero hay algunos factores que podrían afectar la integridad de la datos enviados.
C = Debido a que no se proporcionaron datos significativos, no fue posible evaluar la integridad de los datos enviados.
D = Debido a un factor o una combinación de factores, se debe dar poca credibilidad a los datos enviados.

3.1 Descripción del repositorio ISBSG

ISBSG consta de dos repositorios de datos: de mantenimiento y soporte (*Maintenance and Support Repository (M&S))* y de desarrollo y mejora de software (*Development and Enhancement Repository (D&E)*), así (ISBSG, 2016a):

3.1.1 *Development and Enhancement Repository* (D&E) – R1 de marzo de 2016

El repositorio (D&E), consta de 7518 proyectos de software, desarrollados en 32 países diferentes, con una amplia variedad de tipos de industrias y negocios para su estimación, conocimiento de tendencias, comparación de plataformas e idiomas o evaluación comparativa. El 48% de los proyectos se completaron en la última década. Los datos incluyen (ISBSG, 2016b):

- Detalles de descripción del proyecto.
- Tamaño y atributos de tamaño.
- Esfuerzo y atributos de esfuerzo.
- Defectos.
- Detalles del programa del proyecto.
- Esfuerzo por fase.
- Arquitectura.
- Técnicas.
- Documentación utilizada.
- Detalles de la plataforma.

3.1.2 Información demográfica de los proyectos

Los proyectos pertenecen a 26 países diferentes. Estos son: (29%), Australia (11%), Japón (11%), España (11%), Finlandia (8%), Francia (8%), los Países Bajos (6%), India (4%) y Canadá (4%); (ISBSG, 2016b).

Los principales sectores industriales son: comunicaciones (23%), seguros (18%), manufactura (13%), gobierno (11%), banca (9%), atención médica y de salud (8%) y financiero (6%); (ISBSG, 2016b).

Los principales áreas comerciales son: telecomunicaciones (19%), seguros (9%), banca (9%), transporte / logística (9%), finanzas (9%), fabricación (8%) y ventas y marketing (5%); (ISBSG, 2016b).

En cuanto al tipo de desarrollo: el 67% son proyectos de mejora, el 31% son desarrollos nuevos y el 1% son reconstrucciones (ISBSG, 2016b).

En cuanto al tipo de mercado: el 82% de los proyectos se desarrollaron para uso interno y el 18% para otros usuarios. El 36% se desarrolló internamente y el 64% se subcontrató. En total, el 39% se desarrolla para uso interno (ISBSG, 2016b).

Respecto al tamaño del equipo: 30% de los proyectos tienen hasta 4 personas en el equipo de desarrollo, 30% tienen de 5 a 9 personas, 17% tienen de 10 a 19 personas y el 23% tienen 20 o más personas (ISBSG, 2016b).

3.1.3 *Maintenance and Support Repository* (M&S)

El repositorio mantenimiento y soporte (M&E), consta de aproximadamente 1,100 proyectos, con una variada gama de organizaciones (incluidas las empresas de *Outsourcing* o subcontratación). Los datos incluyen (ISBSG, 2016b):

- Detalles de descripción de la organización
- Detalles de descripción de la aplicación
- Desglose de esfuerzo para actividades de mantenimiento
- Desglose del esfuerzo para actividades de apoyo
- Esfuerzo por tasas de tamaño de la aplicación
- Recuento de defectos
- Plataforma
- Hardware
- Lenguajes de programación utilizados

3.2 Análisis Multivariante

El análisis multivariante es una técnica estadística de síntesis de la información, o reducción de la dimensión (número de variables). Es decir, ante un banco de datos contenido en un conjunto de variables originales, el objetivo será reducirlas en un conjunto más pequeño de variables (factores) perdiendo la menor cantidad de información posible. Es decir, busca si es posible describir con precisión los valores de p variables por un pequeño subconjunto r < p de ellas, reduciendo la dimensión del problema con la menor perdida de información posible (Hair *et al.*, 1999).

La técnica de Análisis de Componentes que encuentra sus orígenes en los ajustes ortogonales por mínimos cuadrados persigue dicho objetivo, dadas *n* observaciones en *p* variables, se analiza si es posible representar adecuadamente esta información con un número menor de variables obtenidas mediante combinación lineal de las originales, de forma que los primeros nuevos componentes expliquen el máximo de la variabilidad total (mayor varianza), mientras que el resto vaya explicando menor cantidad de varianza de forma sucesiva. Además, las sucesivas combinaciones lineales se extraen de manera que no exista correlación entre ellas (Hair *et al.*, 1999; Peña, 2002).

Con variables de alta dependencia, es frecuente que un pequeño número de nuevas variables (menos del 20% de las originales) expliquen la mayor parte (más del 80% de la variabilidad original); (Hair *et al.*, 1999; Peña, 2002).

El objetivo del análisis de componentes principales es, entonces, determinar nuevos factores (componentes principales) que expliquen la mayor cantidad de la variabilidad existente en los datos utilizando, para ello, el menor número de factores posible (Hair *et al.*, 1999; Peña, 2002).

Finalmente, su utilidad es doble. Primero, permite representar en un espacio de dimensión pequeña observaciones de un espacio general *p-dimensional.* Por tanto, es el primer paso para identificar las posibles variables latentes (o no observadas) que generan los datos. Segundo, permite transformar las variables originales, en general correlacionadas, en nuevas variables no correlacionadas en donde no existe correlación lineal, facilitando la interpretación de los datos (Hair *et al.*, 1999; Peña, 2002).

3.3 Aplicación de la técnica de componentes principales:

En este trabajo se utiliza el repositorio *Development and Enhancement Repository (D&E) – R1* de marzo de 2016 (ISBSG, 2016b), una base de datos internacional de proyectos reales de desarrollo de software, con 7518 registros. En total de este repositorio, se seleccionan 15 factores cualitativos principales, relacionados con el ciclo de vida de desarrollo del software y experiencia de la organización en la estimación de proyectos.

Posteriormente, todos estos registros se categorizan de manera cuantitativa y finalmente se analizan mediante el enfoque estadístico del análisis de componentes principales, mediante en el software Minitab 16 Statical Software (Académico), así:

3.3.1 Categorización de las variables

Las siguientes es la descripción de las 15 variables cualitativas principales seleccionadas del repositorio (ISBSG, 2016c) y la categorización cuantitativa propuesta:
Las primeras seis variables que se tienen en cuenta son aquellas que definen el alcance de la actividad del proyecto. Estos factores corresponden a una Categorización binaria (1 o 0): 1, si la variable se incluyó en el proyecto de software. 0, si no se incluyó, así:

- ***CP1: Planeación***
- ***CP2: Especificación***
- ***CP3: Diseño***
- ***CP4: Construcción***
- ***CP5: Pruebas***
- ***CP6: Implementación***
- ***CP7: Tipo de desarrollo,*** este campo describe el tipo de proyecto, así:

Categoría/Descripción

1. Nuevo
2. Mejora
3. Refactorización

- ***CP8: Tamaño relativo,*** basados en *Functional Size Measurement Method (FSM Method)* este campo describe la clasificación funcional del tamaño de proyecto en tamaño relativos, así:

Categoría/Descripción

1. XXS Extra-extra-pequeño => 0 and <10
2. XS Extra-pequeño => 10 and <30
3. S Pequeño => 30 and <100
4. M1 Mediano1 => 100 and <300
5. M2 Mediano2 => 300 and <1000
6. L Grande => 1,000 and < 3,000
7. XL Extra-grande => 3,000 and < 9,000
8. XXL Extra-extra-grande => 9,000 and < 18,000
9. XXXL Extra-extra-extra-grande => 18,000.

- ***CP9: Método de estimación del esfuerzo*** este, campo describe el método utilizado para estimar el esfuerzo, así:

Categoría/Descripción

1. Juicio de Expertos
2. Combinación de modelos con juicio de experto
3. Otros

- ***CP10: Nivel de recursos,*** este campo describe la relación de personas cuyo tiempo se incluye en los datos de esfuerzo de trabajo reportados. Identificando 4 niveles, así:

Categoría/Descripción

1. Esfuerzo del equipo de desarrollo: por ejemplo, equipo del proyecto, administración del proyecto.
2. Apoyo del equipo de desarrollo: por ejemplo, administración de bases de datos, administración de datos, aseguramiento de la calidad, seguridad, soporte de estándares, auditoría y control, soporte técnico.

3. Participación de las operaciones del ordenador: por ejemplo, soporte de software, soporte de hardware, soporte del centro de información, operadores de computadoras, administración de redes.
4. Usuario finales o clientes: por ejemplo, enlaces de usuario, tiempo de formación del usuario, usuarios de aplicaciones y / o clientes.

- ***CP11: Tamaño del grupo:*** este campo define el tamaño máximo del grupo de trabajo para aumentar el número de proyectos trabajados, así:

Categoría/Descripción

1. => 1.55
2. = 1.55 to <2.5
3. 3-4 = 2.5 to <4.5
4. 5-8 = 4.5 to <8.5
5. 9-14 = 8.5 to <14.5
6. 15-20 = 14.5 to <20.5
7. 21-30 = 20.5 to <30.5
8. 31-40 = 30.5 to <40.5
9. 41-50 = 40.5 to <50.5
10. 51-60 = 50.5 to <60.5
11. 61-70 = 60.5 to <70.5
12. 71-80 = 70.5 to <80.5
13. 81-90 = 80.5 to <90.5
14. 91-100 = 90.5 to <100.5
15. 100 + => 100

- ***CP12: Experiencia de contador:*** *este campo describe el número de años de experiencia en el recuento de puntos de función que la persona que hace las veces del contador tenía antes de la medición de tamaño funcional, así:*

Categoría/Descripción

1. Menos de 6 meses
2. Entre 6 meses a 2 años
3. Entre 2 a 5 años
4. Más de 5 años

- ***CP13: Experiencia en gestión de proyectos,*** este campo describe el número de proyectos anteriores de tecnologías de la información (TI) y no (TI) de los cuales el gerente del proyecto fue responsable. Esto es una indicación de la experiencia pasada del gerente del proyecto.

- ***CP14: Cambios de gestor de proyectos***, este campo describe el número de cambios en el administrador del proyecto que ocurrieron durante el mismo.

- ***CP15: Cambios de personal,*** este campo describe el número de personas reemplazadas durante el proyecto por razones tales como enfermedad, renuncia, traslado, licencia de maternidad e incumplimiento del proyecto.

3.3.2 Análisis de componentes principales

Por medio del enfoque estadístico del análisis de componentes principales, realizado en el software Minitab 16 Statical Software (Académico), se buscó encontrar la relación entro los 15 anteriores factores: planeación (CP1), especificación (CP2), diseño (CP3), construcción (CP4), pruebas(CP5), implementación (CP6), tipo de desarrollo (CP7), tamaño relativo (CP8), método de estimación del esfuerzo (CP9), nivel de recursos (CP10), tamaño del grupo (CP11), experiencia de contador (CP12), experiencia en gestión de proyectos (CP13), cambios de gestor de proyectos (CP14), cambios de personal (CP15); es decir, cómo se asocian y se distancian para poder determinar nuevos componentes principales que expliquen la mayor cantidad de variabilidad existente entre los datos utilizando el menor número de factores posibles. La condición para poder prever la mínima pérdida de información de los datos es utilizar la variable de máxima variabilidad. Con este análisis se obtuvieron la **Tabla 6:** matriz de correlación y la **Tabla 7:** matriz de valores propios, así:

Tabla6: Matriz de correlación (Fuente propia).

FACTOR	CP1	CP2	CP3	CP4	CP5	CP6	CP7	CP8	CP9	CP10	CP11	CP12	CP13	CP14	CP15
Planeación	0,369	0,323	-0,159	0,090	0,050	-0,176	0,027	-0,090	-0,007	0,042	0,264	-0,128	-0,116	-0,529	-0,550
Especificación	0,388	0,319	-0,127	0,059	0,028	-0,188	-0,009	-0,125	0,010	-0,015	0,209	-0,146	-0,186	-0,011	0,759
Diseño	0,404	0,299	-0,066	-0,020	0,011	-0,072	-0,020	-0,010	0,053	-0,098	-0,098	0,030	-0,012	0,781	-0,323
Construcción	0,391	0,209	0,049	-0,118	0,013	0,164	0,065	0,234	-0,029	-0,069	-0,421	0,376	0,535	-0,276	0,123
Pruebas	0,313	-0,084	0,325	-0,293	-0,098	0,418	0,056	0,264	-0,114	0,267	-0,104	-0,034	-0,590	-0,050	-0,012
Implementación	0,247	-0,204	0,449	0,044	-0,354	0,066	-0,003	0,075	-0,165	-0,112	0,476	-0,312	0,435	0,073	-0,010
Tipo de desarrollo	0,060	0,063	0,277	0,501	0,511	0,393	-0,478	-0,110	0,061	0,046	0,035	0,001	0,028	0,005	0,003
Tamaño relativo	0,036	-0,011	-0,488	-0,150	-0,472	0,437	-0,509	-0,187	0,131	0,043	0,077	0,015	0,063	-0,021	0,000
Método de estimación del esfuerzo	0,210	-0,328	-0,204	0,071	0,105	-0,192	-0,047	-0,171	-0,405	0,704	-0,125	-0,046	0,191	0,085	0,004
Nivel de recursos	0,030	-0,125	-0,022	-0,683	0,488	-0,158	-0,362	0,122	-0,023	-0,099	0,294	0,050	0,092	0,007	-0,006
Tamaño del grupo	0,166	-0,188	0,428	-0,086	-0,223	-0,405	-0,295	-0,427	0,386	0,020	-0,258	0,164	-0,096	-0,100	-0,021
Experiencia de contador	0,217	-0,390	-0,133	0,233	-0,031	-0,024	0,045	-0,030	-0,253	-0,309	0,280	0,662	-0,219	0,014	-0,002
Experiencia en gestión de proyectos	0,160	-0,297	-0,194	0,241	-0,003	-0,175	-0,076	0,627	0,570	0,167	0,068	-0,055	0,005	0,011	0,018
Cambios de gestor de proyectos	0,227	-0,370	-0,195	0,052	0,104	-0,015	-0,089	-0,013	-0,189	-0,515	-0,435	-0,496	-0,086	-0,091	-0,013
Cambios de personal	0,192	-0,269	-0,101	-0,139	0,258	0,346	0,517	-0,423	0,443	0,027	0,119	-0,051	0,121	0,022	0,004

La **Tabla 7**, incluye la matriz de valores propios asociados con cada componente principal, la proporción de varianza y la varianza explicada acumulada por cada uno de ellos.

Tabla7: Matriz de valores propios (Fuente propia).

COMPONENTE	VALOR PROPIO	% VARIANZA (PROPORCIÓN)	% VARIANZA ACUMULADA
CP1	3,7886	0,253	0,253
CP2	3,0052	0,200	0,453
CP3	1,5242	0,102	0,555
CP4	1,1535	0,077	0,631
CP5	1,0359	0,069	0,700
CP6	0,8117	0,054	0,755
CP7	0,6963	0,046	0,801
CP8	0,6885	0,046	0,847
CP9	0,6061	0,040	0,887
CP10	**0,5207**	**0,035**	**0,922**
CP11	0,3961	0,026	0,948
CP12	0,3156	0,021	0,969
CP13	0,2780	0,019	0,988
CP14	0,1413	0,009	0,997
CP15	0,0382	0,003	1,000

3.3.3 Selección del número de componentes

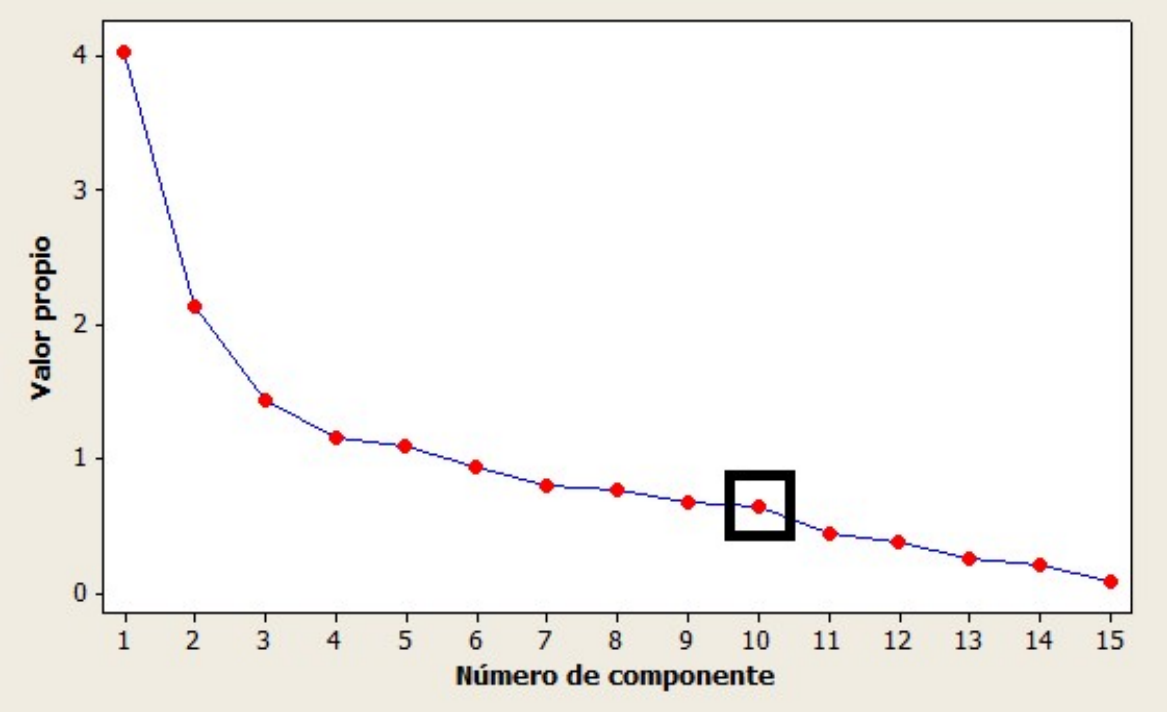
Peña (2002) sugiere las siguientes reglas para seleccionar el número de componentes:

1. “Realizar un gráfico de λi frente a i. Comenzar seleccionando componentes hasta que los restantes tengan aproximadamente el mismo valor de λi. Buscando un “codo” en el gráfico, es decir, punto a partir del cual los valores propios son aproximadamente iguales. El criterio es quedarse con un número de componentes que excluya los asociados con valores pequeños y aproximadamente del mismo tamaño”.
2. “Seleccionar componentes hasta cubrir una proporción determinada de varianza, como el 80 o el 90 por 100. Esta regla es arbitraria y se debe aplicar con cierto cuidado”.

3. “Desechar aquellos componentes asociados con valores propios inferiores a una cota, que se suele fijar como la varianza media, $\sum {\lambda i}/{p}$. En particular, cuando se trabaja con la matriz de correlación, el valor medio de los componentes es 1 y esta regla lleva a seleccionar los valores propios mayores que la unidad”.

Aplicando las anteriores reglas, se realizó la representación gráfica de los componente nuevamente con el software Minitab 16 Statical Software Académico; y en función de los 15 factores iniciales se buscó en la gráfica de sedimentación un punto “codo” donde los valores propios comienzan a ser aproximadamente iguales, se redujeron los factores hasta cubrir una proporción de varianza del 90 por 100 y se seleccionaron los componentes asociados con un valor medio superior a 1, por lo que se decidió emplear 10 componentes principales que se corresponden al (92%) de la varianza acumulada. Como se observa en la **Figura 3.**

Figura 3: Gráfica de sedimentación (Fuente propia).



3.3.4 Interpretación de los componentes principales

- **Componentes de tamaño y forma**

Se analizaron las correlaciones de cada componente principal aplicando las reglas de Peña (2002) y se seleccionaron los valores propios mayores a la unidad interpretando así los factores:

El componente principal *CP1-Planeación*, representa el 92,2% de la variabilidad total. De esta manera, la mayor parte de la estructura de datos de puede capturar en 10 dimensiones subyacentes. Los componentes principales restantes: *CP11 tamaño del grupo (94,8%), CP12 experiencia de contador (96,9%), CP13 experiencia en gestión de proyectos (98,8%), CP14 cambios de gestor de proyectos (99,7%) y CP15 cambios de personal (100%),* explican en menor proporción la variabilidad y por tanto, podrían tener menor importancia en el desarrollo de proyectos de software.

Este primer componente tiene todas sus coordenadas del mismo signo por tanto existe una alta correlación positiva entre todas las variables se puede interpretar como un promedio ponderado de todas las variables, o un factor global de “tamaño”. Los demás componentes se pueden interpretar como factores de “forma” y cumplen con que tienen coordenadas positivas y negativas, que implican que contraponen unos grupos de variables frente a otros.

En cuanto a la varianza es de 3,7886 y explica el 25,3% de la variabilidad de los datos. Representan el tamaño global de un proyecto de desarrollo de software tanto en el ciclo de vida del desarrollo: *planeación (0,369), especificación (0,388), diseño (0,404), construcción (0,391), pruebas (0,313) e implementación (0,247),* como otros factores importantes a tener en cuenta en un proyecto de software como el *tipo de desarrollo (0,060), el tamaño del proyecto (0,036), el método de estimación del esfuerzo (0,210) , el nivel de recursos (0,030) el tamaño del grupo (0,166), la experiencia del contador (0,217), la experiencia en gestión de proyectos (0,160), los cambios de gestor del proyecto (0,227) y los cambios de personal (0,192);* porque los coeficientes de estos factores tienen todos signo positivo y se alejan del cero.

Es posible asimilar los valores positivos de correlación con aquellos proyectos de desarrollo de software que involucran e interrelacionan tanto los factores inherentes al desarrollo de software como a la organización, lo que demuestra la influencia e importancia tanto de los aspectos ingenieriles propios del ciclo de vida del desarrollo de software como todos aquellos factores que involucran la toma de decisiones organizacionales y la interacción del grupo de trabajo.

El segundo componente *CP2-Especifiación*, tiene una varianza de 3,0052 y explica el 45,3% de la variabilidad de los datos. Este componente tiene correlación positiva de valor absoluto mayor con los factores de *planeación (0,323) y especificación (0,319); y correlación negativa de valor absoluto mayor con los factores de método de estimación (-0,328), experiencia del contador (-0,390) y cambios del gestor de proyecto (-0,370).* Estos valores se podrían asimilar con proyectos de desarrollo de software con sólida planeación y especificación de requisitos, pero poca solidez u apoyo de los métodos de estimación (juicios de expertos, combinación de juicio de expertos con modelos u otros), con baja experiencia del contador y frecuentes cambios del gestor de proyectos.

El tercer componente *CP3-Diseño*, tiene una varianza de 1,5242 y explica el 55,5% de la variabilidad de los datos. Este componente tiene correlación positiva de valor absoluto mayor con los factores de *pruebas (0,325), implementación (0,449) y tamaño del grupo (0,428);* y correlación negativa de valor absoluto mayor con el factor de *tamaño relativo (-0,488), l*o que se asimila con proyectos de desarrollo de software que llegaron hasta su implementación y con fortaleza en el desarrollo de pruebas, con grupos grandes de trabajo en proyectos de tamaño pequeño.

El cuarto componente *CP4-Construcción*, tiene una varianza de 1,1535 y explica el 63,1% de la variabilidad de los datos. Este componente sólo tiene correlación positiva de valor absoluto mayor con el factor de *tipo de desarrollo (0,501)* y correlación negativa de valor absoluto mayor con el factor de *nivel de recursos (-0,683),* lo que se asimila con diversos proyectos de desarrollo de software (nuevos, mejoras o refactorización) y bajo nivel de recursos (esfuerzo del equipo de desarrollo, apoyo del equipo de desarrollo y participación de las operaciones del ordenador).

El quinto componente *CP5-Pruebas,* tiene una varianza de 1,0359 y explica el 70% de la variabilidad de los datos. Este componente tiene correlación positiva de valor absoluto mayor con los factores de *tipo de desarrollo (0,511) y nivel de recursos (0,488);* y correlación negativa de valor absoluto mayor con los factores de *implementación (-0,354) y tamaño relativo (-0,472),* lo que se asimila proyectos de desarrollo de tamaño pequeño, con capacidad de recursos (esfuerzo del equipo de desarrollo, apoyo del equipo de desarrollo y participación de las operaciones del ordenador) pero con poca capacidad de implementación.

El sexto componente *CP6-Implementación,* tiene una varianza de 0,8117 y explica el 75,5% de la variabilidad de los datos. Este componente tiene correlación positiva de valor absoluto mayor con los factores *de pruebas (0,418), tipo de desarrollo (0,393), tamaño relativo (0,437) y cambios de personal (0,346)*; y correlación negativa de valor absoluto mayor con el factor de *tamaño del grupo (-0,405),* lo que se asimila con grandes proyectos de desarrollo de software, que se han apoyado en algún método de estimación (juicios de expertos, combinación de juicio de expertos con modelos u otros) y el desarrollo de pruebas, con pequeños grupos de trabajo y frecuentes cambios de personal.

El séptimo componente *CP7-Tipo de desarrollo*, tiene una varianza de 0,6963 y explica el 80,1% de la variabilidad de los datos. Este componente tiene correlación positiva de valor absoluto mayor con el factor de *cambios de personal (0,517);* y correlación negativa de valor absoluto mayor con los factores de *tipo de desarrollo (-0,478), tamaño relativo (-0,509) y nivel de recursos (-0,362),* lo que se asimila con proyectos de desarrollo de software de tamaño pequeño, con poca variación en el tipo (nuevo desarrollo, mejora o refactorización), con poca capacidad de recursos (esfuerzo del equipo de desarrollo, apoyo del equipo de desarrollo y participación de las operaciones del ordenador) y altos cambios de personal.

El octavo componente *CP8-Tamaño relativo*, tiene una varianza de 0,6885 y explica el 84,2% de la variabilidad de los datos. Este componente sólo tiene correlación positiva de valor absoluto mayor con el factor de *experiencia del gestor de proyectos (0,627);* y correlación negativa de valor absoluto mayor con los factores de *tamaño del grupo de trabajo (-0,427) y cambios de personal (-0,423),* lo que se asimila con proyectos de

desarrollo de software en los que el gestor del proyecto (TI y no TI) tiene alta experiencia, con grupos de trabajo pequeños y pocos cambios de personal.

El noveno componente *CP9-Método de estimación de esfuerzo*, tiene una varianza de 0,6061 y explica el 88,7% de la variabilidad de los datos. Este componente tiene correlación positiva de valor absoluto mayor con los factores de *tamaño del grupo (0,386), experiencia del gestor de proyectos (0,570) y cambios de personal (0,443);* y correlación negativa de valor absoluto mayor con el factor de método de *estimación del esfuerzo (-0,405),* lo que se asimila con proyectos de desarrollo de software con grupos de trabajo grandes, con alta experiencia del gestor de proyectos, frecuentes cambios de personal y poca utilización de métodos de estimación (juicio de expertos, combinación de juicio de expertos con modelos u otros).

El décimo componente *CP10-Nivel de recursos*, tiene una varianza de 0,5207 y explica el 92,2% de la variabilidad de los datos. Este componente tiene correlación positiva de valor absoluto mayor con el factor de *método de estimación del esfuerzo (0,704);* y correlación negativa de valor absoluto mayor con los factores de *experiencia del contador (-0,309) y cambios del gestor de proyectos (-0,515).* Estos son proyectos de desarrollo de software con un fuerte componente de apoyo en algún método de estimación (juicio de expertos, combinación de juicio de expertos con modelos u otros) que cuentan con experiencia previa en gestión de proyectos tanto de TI (Tecnologías de la información) como no TI, con pocos cambios de personal y de gestor de proyectos a pesar de la baja experiencia del contador de puntos de función.

4. Conclusiones y recomendaciones

4.1 Conclusiones

La estimación del esfuerzo de desarrollo de software se traduce en un asunto de dinero, que el tiempo y la premura en obtener el producto afectan. Por ello, la academia y la industria del software a nivel internacional y local, desde hace varias décadas, estudian el problema de la estimación. Sin embargo, aún no encuentran un consenso y una mejora en el proceso y la precisión de la estimación. En la literatura se encuentra diversas técnicas y modelos de estimación divididas de manera principal en modelos algorítmicos y no algorítmicos.

A partir de los casos de estudio analizados se encontró que entre los métodos más usados para la estimación de esfuerzos se encuentra en primer lugar el juicio de expertos, en segundo lugar los métodos algorítmicos y en tercer lugar la mejora de procesos. Esto permite pensar que una combinación de juicio de expertos con un método algorítmico permitiría mejorar la precisión de los resultados de la estimación de esfuerzos.

En la revisión de la literatura se encuentran diferentes casos de estudio que muestran que el juicio de expertos es la técnica más empleada para pronosticar esfuerzos de sus proyectos de desarrollo de software, pero que poco se usa en combinación con otras técnicas. En América Latina, y de manera específica en Colombia, la industria del software la dominan pequeñas empresas dedicadas al desarrollo de proyectos de poco tamaño con duraciones entre 1 a 6 meses y expuestos por los pocos tiempos de desarrollo al caos, la incertidumbre y la imprevisibilidad, dando como resultados proyectos que fracasan por la subestimación o la sobreestimación de los mismos. Por tanto, se podrían beneficiar con técnicas que se ajusten mejor a sus particularidades.

Existen múltiples factores identificados en los casos de estudio reportados, aunque no existe de forma explícita un consenso sobre la totalidad de estos; sí se identifican como importantes la complejidad del proyecto, el tamaño del proyecto y la disponibilidad de información histórica.

Se identificaron y caracterizaron los factores de mayor incidencia a partir de información existente de proyectos de desarrollo de software, que se incluyen en el repositorio ISBSG. El análisis de estos factores permitió identificar como los factores más importantes a: planeación (0,369), especificación (0,388), diseño (0,404), construcción (0,391), pruebas (0,313), implementación (0,247), tipo de desarrollo (0,060), el tamaño del proyecto (0,036), el método de estimación del esfuerzo (0,210) , el nivel de recursos (0,030) el tamaño del grupo (0,166), la experiencia del contador (0,217), la experiencia en gestión de proyectos (0,160), los cambios de gestor del proyecto (0,227) y los cambios de personal (0,192).

Estos factores evidencian la sobreestimación y subestimación a la que se enfrentan las empresas locales de tamaño pequeño y muestran, que los factores relacionados con el ciclo de vida del desarrollo de software (planeación, especificación diseño, construcción y pruebas) y la experiencia de la organización, tipo de desarrollo, tamaño del proyecto, método de estimación de esfuerzo, nivel de recursos, tamaño del grupo, experiencia del contado, experiencia en gestión de proyectos, cambios del gestor de proyecto y cambios de personal, inciden mayormente en los resultados de la estimación de esfuerzo.

4.2 Recomendaciones

Este Trabajo Final de Maestría, deja en conocimiento unos factores principales en la estimación de esfuerzos en el desarrollo de software relacionados con el ciclo de desarrollo y la experiencia de la organización que afectan los resultados de la estimación de esfuerzos, identificados mediante el caso de estudio del repositorio ISBSG.

En la búsqueda de mejorar la precisión de la estimación, es posible aplicar estos factores que afectan la estimación en el desarrollo de nuevos modelos de estimación de esfuerzo, más ajustado a las necesidades de los proyectos especiales de software de las empresas locales.

Esta información requiere mayor análisis, que se puede realizar partiendo de la implementación de un análisis de sensibilidad previo de los datos para luego aplicar técnicas estadísticas y futuros análisis de componentes principales.

5. Referencias

Albrecht, A. J. (1979). Measuring application development productivity. In IBM Corporation (Ed.), *IBO Conference on Application Development* (pp. 83–92).

Almache C., M. G., Raura, G., Ruiz R., J. A., y Fonseca C., E. R. (2015). Modelo neuronal de estimación para el esfuerzo de desarrollo en proyectos de software (MONEPS). *Revista Latinoamericana de Ingeniería de Software*, *3*(3), 148–154.

Aparna, J., Garg, R., y Bansal, A. (2015). Review of effort estimation model - a survey. *International Journal of Research y Development in Technology and Management Science - Kailash*, *22*(3), 32–39.

Attarzadeh, I., y Hock Ow, S. (2010). A novel soft computing model to increase the accuracy of software development cost estimation. *IEEE Transactions on Software Engineering*, *3*, 603–607.

Ayyildiz, M., Kalipsiz, O., y Yavuz, S. (2008). A metric-set and model suggestion for better software project cost estimation. *International Journal of Research and Scientific Innovation (IJRSI)*, *2*(11), 167–172.

Boehm, B. (1981). *Software engineering economics*. (Prentice Hall, Ed.) (1st ed.). New Jersey.

Boehm, B. (2000). COCOMO II Model definition manual. *The American Journal of Tropical Medicine and Hygiene*, (v 2.1), 87.

Boehm, B., Clark, Horowitz, Brown, Reifer, Chulani, … Steece, B. (2000). *Software cost estimation with COCOMO II*. (Prentice Hall, Ed.) (1st ed.). New Jersey.

Dejaeger, K., Verbeke, W., Martens, D., y Baesens, B. (2012). Data mining techniques for software effort estimation: a comparative study. *IEEE Transactions on Software Engineering*, *38*(2), 375–397.

Del Valle Roque, D., Cueto Ible Eduardo, E., y Navarro Guerrero, P. E. (2014). *Diseño de métricas para calcular el costo en el proceso de desarrollo de software*. Universidad de Camagüey.

Ferreira, L., Gheisa, L., Lío, D. G., Alberto, L., Domínguez, Q., y Antón, J. (2014). Estimación del esfuerzo en proyectos de software utilizando técnicas de inteligencia artificial. *Revista Cubana de Ciencias Informáticas*, *8*(4), 20.

Gharehchopogh, F. S., Maleki, I., y Reza Khaze, S. (2014). A novel particle swarm optimization approach for software effort estimation. *International Journal of Academic Research*, *6*(2), 69–76.

Hair, J. F., Rolph E., A., Ronald L., T., y Black, W. C. (1999). *Análisis multivariante*. (P. H. Iberia, Ed.) (5th ed.). Madrid.

Hughes, R. T. (1996). Expert judgement as an estimating method. *Information and Software Technology*, *38*(2), 67–75.

ISBSG. (2016a). ISBSG Delivering IT Confidence. Retrieved from http://isbsg.org/

ISBSG. (2016b). ISBSG Data Demographics D&E Repository Release 2016 R1, *1*(March), 1–27.

ISBSG. (2016c). Field Descriptions ISBSG D&E Repository. Australia.

Jørgensen, M. (2004). A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, *70*(1–2), 37–60.

Jørgensen, M. (2007). Forecasting of software development work effort: evidence on expert judgement and formal models. *International Journal of Forecasting*, *23*(3), 449–462.

Jørgensen, M., y Moløkken-Østvold, K. (2004). Reasons for software effort estimation error: Impact of respondent role, information collection approach, and data analysis method. *IEEE Transactions on Software Engineering*, *30*(12), 993–1007.

Jorgensen, M., y Shepperd, M. (2007). A systematic review of software development cost estimation studies. *IEEE Transactions on Software Engineering*, *33*(1), 33–53.

Kaczmarek, J., y Kucharski, M. (2004). Size and effort estimation for applications written in Java. *Information and Software Technology*, *46*(9), 589–601.

Kashyap, D., Shukla, D., y Misra, A. K. (2014). Refining the use case classification for use case point method for software effort estimation. *Association of Computer Electronics and Electrical Engineers*, 183–191.

Kaur, T., y Singh, J. (2015). A hybrid model for the enhancement in software effort estimation. *International Journal of Scientific y Engineering Research*, *6*(7), 619–624.

Kaushik, A., Soni, A. K., y Soni, R. (2012). An adaptive learning approach to software cost estimation. In IEEE Transactions on Software Engineering (Ed.), *National Conference on Computing and Communication Systems (NCCCS)* (pp. 1–5). Durgapur, West Bengal, India.

Khatibi, V., y Jawawi, D. N. . (2010). Software cost estimation methods : a review. *Journal of Emerging Trends in Computing and Information Sciences*, *2*(1), 21–29.

Macdonell, S. G., y Shepperd, M. J. (2003). Combining techniques to optimize effort predictions in software project management. *The Journal of Systems and Software*, *66*, 91–98.

Maleki, I., Ghaffari, A., y Masdari, M. (2014). A new approach for software cost estimation with hybrid genetic algorithm and ant colony optimization. *International Journal of Innovation and Applied Studies*, *5*(1), 72–81.

Mansor, Z., Kasirun, Z. M., Yahya, S., y Arshad, N. H. H. (2011). Current practices of software cost estimation technique in malaysia context. In *Informatics Engineering and Information Science* (pp. 566–574). Springer.

Mendes, E. (2008). *Cost estimation techniques for web projects*. (IGI PubIIshInG, Ed.). New York.

Mendes, E. (2012). Improving software effort estimation using an expert-centred approach. In M. Winckler, P. Forbrig, y R. Bernhaupt (Eds.), *Human-Centered Software Engineering: 4th International Conference, HCSE 2012,* (pp. 18–33). Toulouse, France, Heidelberg: Springer Berlin Heidelberg.

Moløkken-Østvold, K., Jørgensen, M., Tanilkan, S. S., Gallis, H., Lien, A. C., y Hove, S. E. (2004). A survey on software estimation in the norwegian industry. *10th International Symposium on Software Metrics (METRICS'04). IEEE Computer Society*, 208–219.

Morales, O., y Cooke, R. M. (2010). Introducción al modelo clásico de juicio estructurado de expertos : breve recuento del pasado y una aplicación reciente. *CIENCIA Ergo Sum*, *16*(3), 309–318.

Nasir, M., y Ahmad, H. F. (2006). An empirical study to investigate software estimation trend in organizations targeting CMMISM. In *5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering,Software Architecture and Reuse (ICIS-COMSAR'06)* (pp. 38–43).

Páez, I. (2012). *Estudio empirico del estado actual de la estimación de software en pymes de Colombia*. Universidad EAFIT.

Pandey, P. (2013). Analysis of the techniques for software cost estimation. In *2013 Third International Conference on Advanced Computing y Communication Technologies.* (pp. 16–19). IEEE.

Peláez Valencia, L. E., Cardona Benjumea, L., y Toro Lazo, A. (2011). Estado del arte que soporta el proceso de desarrollo de software en las PYMES Colombianas: una mirada desde las organizaciones nacionales que tienen que ver con la disciplina. *Entre Ciencia E Ingeniería. Universidad Católica de Pereira*, *5*(10), 93–107.

Peña, D. (2002). *Análisis de datos multivariantes*. (Mcgraw Hill / Interamericana de España, Ed.) (1st ed.). Madrid.

Pérez, A. L., González, L., Duque, A., Millane, F., y Ospina, G. (2006). Modelo para la estimación temprana de esfuerzo en proyectos de software, incorporando información de proyectos similares. *Revista Avances en Sistemas e Informática*, *3*(2), 65–70.

Pesado, P., Esponda, S., Pasini, A., Boracchia, M., Swaels, M., y Bertone, R. (2014). Buenas prácticas para la mejora de procesos en el desarrollo de software y en procesos de gestión . Aplicación en Pymes. In *WICC 2014 XVI Workshop de Investigadores en Ciencias de la Computación* (pp. 514–519). Argentina.

Pino, F., García, F., y Piattini, M. (2006). Revisión sistemática de mejora de procesos software en micro , pequeñas y medianas empresas. *Revista Española de Innovación Calidad E Ingeniería Del Software REICIS*, *2*(1), 6–23.

Pushkar, S., y Kumari, S. (2013). Performance analysis of the software cost estimation methods : a review. *International Journal of Advanced Research in Computer Science and Software Engineering*, *3*(7), 229–238.

Putnam, L. H. (1978). A general empirical solution to the macro software sizing and estimating problem. *IEEE Transactions on Software Engineering*, *SE-4*(4), 345–361.

Rajkumar, G., y Alagarsamy, K. (2013). A systematic review of cost estimation models. *Journal of Global Research in Computer Science*, *4*(5), 33–36.

Ramírez, A., y Orantes, S. (2011). Prototipo de una herramienta de apoyo para la estimación de costos, en la etapa de desarrollo de un proyecto de Software. *Revista Digital Universitaria.Universidad Nacional Autónoma de México*, *12*(6), 1–15.

Robiolo, G. (2009). *Transacciones, objetos de entidad y caminos: métricas de software basadas en casos de uso, que mejoran la estimación temprana de esfuerzo*. Universidad Nacional de la Plata.

Robiolo, G., y Santos, S. (2013). ¿Es posible superar la precisión basada en el juicio de expertos de la estimación de esfuerzo de productos de software? In *CIbSE 2013 | X Workshop Latinoamericano Ingeniería de Software Experimental | ESELAW 2013* (pp. 48–60). Buenos Aires, Argentina.

Robiolo, G., y Santos, S. (2014). Estimación de proyectos de software pequeños basada en el juicio de expertos: un caso de estudio. In *ASSE 2016, 17º Simposio Argentino de Ingeniería en Software* (pp. 39–50). Buenos Aires, Argentina.

Ruíz Constanten, Y., y Cordero Morales, D. (2013). Estimación en proyectos de software integrando los métodos de Boehm y Humphreyc. *Revista Cubana de Ciencias Informáticas*, *7*(3), 23–36.

Salvetto De León, P, F. (2006). *Modelos automatizables de estimación muy temprana del tiempo y esfuerzo de desarrollo de sistemas de información*. Universidad Politécnica de Madrid.

Santos, S. A. (2014). *Estimación de proyectos de software pequeños basada en el juicio de expertos*. Universidad Nacional de La Plata.

Saraç, Ö. F., y Duru, N. (2013). A novel method for software sffort estimation: estimating with boundaries. *IEEE Transactions on Software Engineering*, 1–5.

Suri, P. K., y Ranjan, P. (2012). Comparative analysis of software effort estimation techniques. *International Journal of Computer Applications*, *48*(21), 12–19.

Tailor, O., Saini, J., y Rijwani, P. (2014). Comparative analysis of software cost and effort estimation methods : a review. *International Journal of Computer Science and Mobile Computing*, *3*(4), 1364–1374.

Torres Ricaurte, D. M. (2013). *Un método para medir la productividad del equipo de pruebas en la estimación del esfuerzo de pruebas de software*. Universidad Nacional de Colombia, Medellín.

Velásquez H., J. D., Dyner R., I., y Souza, R. C. (2006). Políticas para la integración del juicio experto y los pronósticos estadísticos en el marco organizacional. *Estudios Gerenciales*, *133*, 131–150.

Villareal, D. (2007). *Estimación de esfuerzo y costo en la producción de software hecho en Venezuela: estudio de casos.* Universidad de los Andes Mérida (ULA), Mérida, Venezuela.

Yahya, M. A., Ahmad, R., y Lee, S. P. (2008). Effects of software process maturity on COCOMO II effort estimation from CMMI perspective. In *IEEE International Conference on Research, Innovation and Vision for the Future, 2008. RIVF 2008* (pp. 255–262).

Yang, D., Wang, Q., Li, M., Yang, Y., Ye, K., y Du, J. (2008). A survey on software cost estimation in the chinese software industry. *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. Kaiserslautern, Germany*, 253–262.