UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 5, Laboratorio de Gestión de Proyectos

Facultad 3

Estimación del esfuerzo en proyectos a partir de puntos de función y técnicas de soft computing

Trabajo final presentado en opción al título de

Máster en Gestión de Proyectos Informáticos

Autor: Ing. Liannis Soria Barreda

Tutor: Dr. C Pascual Verdecia Vicet

Ciudad de La Habana, Junio de 2015

AGRADECIMIENTOS

A mi tutor desde el pregrado, Julio Cesar Diaz Vera, ¿llegaremos al doctorado?

A todos los que preguntaban ¿cómo va la tesis?, ¿en qué te puedo ayudar?

Al equipo de la maestría por darme esta oportunidad

Al Dr. C Pascual Verdecia por las minuciosas revisiones realizadas

DEDICATORIA

A mi hijo, este esfuerzo es por ti

A mi familia por haberme apoyado siempre sin importar las distancias

A Dios por haberme dado la vida y con esta la capacidad para lograr lo que me proponga, superando obstáculos y llegando a las metas propuestas

DECLARACIÓN JURADA DE AUTORÍA Y AGRADECIMIENTOS

Declaro por este medio que yo Liannis Soria Barreda, con carné de identidad 84111826737, soy el autor principal del trabajo final de maestría Estimación del esfuerzo en proyectos a de puntos de función y técnicas de *soft computing*, desarrollada como parte de la Maestría en Gestión de Proyectos Informáticos y que autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Y para que así conste, firmo la presente declaración jurada de autoría en Ciudad de La Habana a los 26 días del mes de junio del año 2015.

_____

Nombre y Firma del maestrante

RESUMEN

La estimación de software a partir del método Puntos de Función se ha popularizado en los últimos años. A pesar de los buenos resultados que se alcanzan con este método, aún existen dificultades que afectan la exactitud de las estimaciones, dentro de estas dificultades se destacan los "límites bruscos". La presente investigación se propone definir un algoritmo de estimación del esfuerzo a partir de puntos de función y técnicas de *soft computing* que contribuya a mejorar la precisión de las estimaciones para la gestión del alcance en el proceso de desarrollo de software. Se obtuvieron como resultados de esta investigación dos algoritmos de estimación del esfuerzo a partir de puntos de función, utilizando para el primero el modelo de representación lingüística de 2-tuplas y para el segundo un Sistema de Inferencia Borroso. Para validar los resultados se utilizó una estrategia de investigación experimental basada en tres casos de estudio; donde los resultados alcanzados con el algoritmo que utiliza el modelo de representación lingüística de 2-tuplas mostraron ser más cercanos a la realidad que el resto de las estimaciones realizadas. La investigación realizada permitió obtener un método que mejora la precisión de la estimación del proceso de desarrollo de un software a partir de puntos de función y técnicas de *soft computing*, lo que contribuye a realizar planificaciones más cercanas a la realidad.

Palabras clave: puntos de función, límites bruscos, sistema de inferencia borroso, representación lingüística

Abstract

The software estimation using Function Point method has become popular in recent years. Despite the good results achieved with this method, there are difficulties that affect the accuracy of the estimates, among these difficulties highlight the "rough limits". This research aims to define an algorithm for estimating the effort using function points and soft computing techniques to help improve the accuracy of estimates for scope management in the software development process. Was obtained as results of this research two algorithms for estimating the effort using function points, for the first one was used the model of linguistic representation of 2-tuples and for the second was used a Fuzzy Inference System. To validate the results was used an experimental research strategy based on three case studies; where the results achieved with the algorithm that used the model of linguistic representation of 2-tuples proved to be closer to reality than the rest of the estimates. The investigation yielded a method that improves estimation accuracy of the process of developing software using function points and soft computing techniques, which helps to perform closer planning to reality.

Keywords: *function point, rough limits, Fuzzy Inference System, linguistic representation*

ÍNDICE

INTRODUCCIÓN

Antecedentes y situación problémica

El mundo con el transcurso del tiempo se ha tornado cada vez más competitivo, por lo que los servicios y/o productos que se ofertan a los clientes pueden marcar la diferencia entre permanecer en el mercado o desaparecer. Como herramienta efectiva para las empresas, que les permite obtener una ventaja competitiva, ha surgido la figura del proyecto. Los proyectos se inician por dos causas fundamentales, para aprovechar una oportunidad de negocio o para darle solución a una situación determinada en una empresa. [Neil 2006]

Al término de proyecto se le pueden asociar una gran variedad de conceptos proporcionados por un gran número de autores reconocidos a nivel mundial. Desde el año 1995, Rafael De Heredia define al proyecto como “una combinación de recursos humanos y no humanos, reunidos en una organización temporal para obtener un propósito determinado” [De Heredia 1995]. Diez años más tarde, Alberto Domingo Ajenjo define al proyecto como “el conjunto de actividades planificadas, ejecutadas y supervisadas que con recursos finitos tiene como objetivo crear un producto o servicio único” [Domingo Ajenjo 2005].

En años más recientes, el Project Management Institute (PMI) define al proyecto como “un esfuerzo temporal que se lleva a cabo para crear un producto, servicio o resultado único” [PMI 2009]. Por último, en la norma ISO 21500 se define al proyecto como “un conjunto de procesos que consta de actividades coordinadas y controladas con fecha de inicio y fin, que se llevan a cabo para lograr los objetivos del proyecto” [ISO 2012]. Todas las definiciones analizadas anteriormente señalan el término de temporal y único para un proyecto, así como la consecución de los objetivos trazados para el proyecto emprendido.

El uso creciente de las Tecnologías de la Información y la Comunicación (TIC) ha tenido un gran impacto en el mundo de hoy y sobre todo en el desarrollo de un proyecto sin importar su naturaleza en una organización. A partir del año 1990 se comienza a vivir la era de la información, donde el software comenzó a tener un papel protagónico en todas las esferas de la vida. Esta situación ha implicado que los esfuerzos llevados a cabo para lograr una gestión exitosa de los proyectos de desarrollo de software, hayan ido en ascenso. En la actualidad, según el Informe del Caos del Standish Group publicado en el 2013, todavía el 18 % de los proyectos de desarrollo de software fracasan y el 43 % son modificados debido a que sobrepasan su costo inicial, terminan después de las fechas acordadas o no entregan todas las funcionalidades requeridas, por lo que sólo el 39 % de los proyectos tuvieron éxito en el año 2012 [Manifesto 2013].

A pesar de registrarse un incremento de los proyectos que tuvieron éxito respecto al estudio anterior en un 2 %, donde el punto más bajo fue registrado en el año 2004 con un 29 % de proyectos exitosos, estas cifras continúan siendo bajas para la industria y representan pérdidas para las

empresas que se dedican al desarrollo de software. En el informe presentado por el Standish Group en [Manifesto 2013] sobre los resultados que se obtienen en los proyectos de desarrollo de software, se definen los diez factores que deben ser asegurados para que un proyecto sea exitoso. Estos diez factores fueron definidos a partir del análisis de los datos recogidos desde el año 2002 en la industria de desarrollo de software por este grupo; lo que comprende alrededor de 50 000 proyectos desarrollados por empresas que se encuentran ubicadas principalmente en los Estados Unidos con un 60 % y en Europa con un 25 %. En la figura 1 se muestran los diez factores definidos en este informe, así como la cantidad de puntos de éxito para representar el impacto del mismo en el éxito del proyecto, donde la sumatoria de estas diez cantidades es igual a 100.

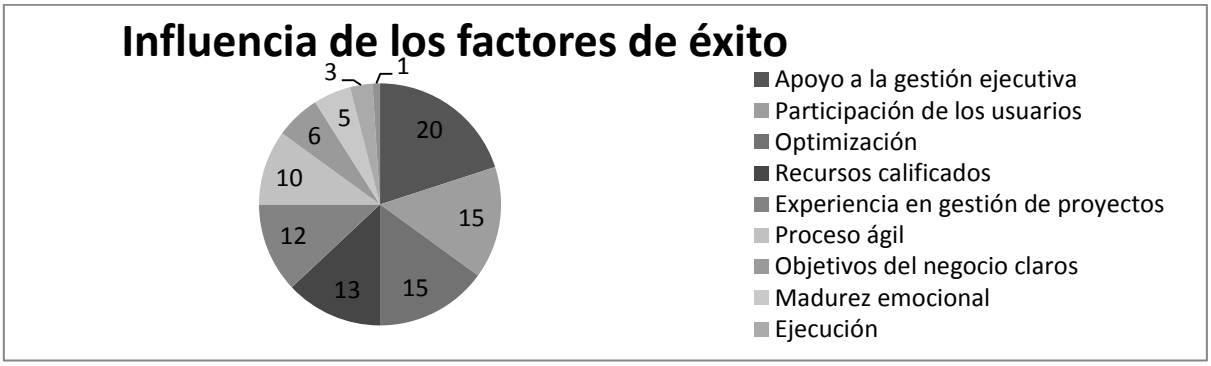


Figura 1. Puntos asociados a los factores de éxito para un proyecto de desarrollo de software según el Standish Group [Manifesto 2013]

Teniendo en cuenta los diez factores definidos por el Standish Group, se puede decir que el éxito o el fracaso de un proyecto, viene dado por la correcta definición y gestión del mismo. El fracaso de un proyecto representa pérdida de dinero y tiempo para una empresa; por lo que se vuelve obligatorio para las empresas mejorar los procesos que llevan a cabo en torno a los proyectos que ejecutan.

Para lograr mejorar de forma objetiva cualquier proceso de desarrollo de un producto es imprescindible medir el proceso y el producto que se elabora. En caso contrario se tomarían decisiones basadas en evaluaciones subjetivas, lo que podría resultar en el fracaso del proyecto. Las mediciones permiten establecer una línea base del proceso que se realiza, a partir de las cuales se establecen un conjunto de indicadores para guiar el proceso. [Salazar-B 2009]

Una buena parte de la comunidad ubica en las malas estimaciones la razón fundamental de que los proyectos de software excedan los parámetros establecidos en su concepción. La estimación constituye una de las tareas más difíciles a realizar en la gestión de un proyecto de software; ya que es un proceso que se lleva a cabo durante todo el ciclo de vida del software y que utiliza valores con niveles elevados de abstracción en sus inicios, para luego refinarlos a medida que progresa el desarrollo y se generan otras informaciones. A pesar de que se han desarrollado numerosas técnicas de estimación de esfuerzo para proyectos de software, no existe un consenso sobre cuál utilizar en cada situación.

Debido a los problemas que han sido detectados en las técnicas de estimación de esfuerzo, varios autores han propuesto realizarle modificaciones con el objetivo de obtener estimaciones más precisas. Por lo que la selección de la técnica de estimación a utilizar en una organización, se convierte en una decisión complicada a tomar, la cual dependerá en gran medida de sus características propias y la experiencia acumulada en la industria de desarrollo de software. En la Universidad de las Ciencias Informáticas (UCI), debido al objetivo propuesto de obtener el nivel 2 del Modelo Integrado de Capacidad y Madurez (CMMI, por sus siglas en inglés) para certificar la calidad de su proceso de desarrollo de software, se decidió desarrollar en el año 2009 un método propio de estimación que se ajustara a sus necesidades.

El método de estimación creado en la UCI se basó en el funcionamiento de los métodos que contaban con mayor reconocimiento internacional en ese momento. Por lo que se obtuvo un método matemático que combinaba algunas de las actividades y procesos de las técnicas de estimación existentes, de forma tal que se adaptaran a las características específicas de la universidad. Este método de estimación fue institucionalizado en la universidad, por lo que es utilizado en la actualidad por todos los proyectos de desarrollo de software que se llevan a cabo en la UCI. Se debe señalar que este método de estimación tampoco se encuentra exento de las debilidades que padecen el resto de los métodos de estimación que existen. Por lo que todavía se trabaja en la universidad en algunas modificaciones que permitan mejorar sus resultados, sin desecharse la posibilidad de utilizar un método diferente que proporcione estimaciones más precisas.

La inclusión del método de estimación de Puntos de Función como ejemplo en el *CMMI for Development* para la estimación de proyectos de software [CMMI 2010], ha permitido elevar la reputación de este método de estimación en los últimos años y por consiguiente su popularidad. Esta técnica tiene como objetivo medir el tamaño de cualquier software en términos orientados a la entrega que se realiza al usuario final, basándose en los requisitos funcionales. Esta medición se realiza independientemente de la plataforma de desarrollo utilizada [Busquelle 2010]. Además de que puede ser utilizada para estimar casos de pruebas, así como para ayudar a entender el crecimiento de los proyectos y para desarrollar un estándar de establecimiento de métricas.

En el año 2012 el grupo consultor DCG, publicó un informe sobre la utilización de los métodos utilizados en la industria para medir el software, donde se evidencia el auge alcanzado por el método Puntos de Función en todo el mundo. Según este informe, el método Puntos de Función alcanzó un 34 % de utilización en las compañías encuestadas, mientras que el resto de los métodos de estimación oscilaron entre un 13 % y un 5 % de utilización [DCG 2013]. Además, en el último informe publicado a inicios del 2015 por la Asociación de Auditoría y Control de Sistemas (ISACA, por sus siglas en inglés), sobre el uso de las métricas en los Estados Unidos, la métrica Puntos de Función logró ubicarse en la cima con un 39 % de utilización, seguido por las líneas de código con un 21 % [Henderson 2015].

El método Puntos de Función consiste en identificar los componentes del sistema en términos de transacciones y grupos de datos lógicos relevantes al usuario; se consideran cinco parámetros básicos: entrada externa, salida externa, consulta externa, archivo lógico interno y archivo de interfaz externo. A estos componentes se les asigna un número de puntos de función teniendo en cuenta su tipo y complejidad. Luego, a la sumatoria de estos puntos, denominada puntos de función sin ajustar, se le aplica un factor de ajuste teniendo en cuenta las características generales del sistema a través de valoraciones subjetivas. [Duran 2003]

A pesar de la popularidad alcanzada por el método Puntos de Función, todavía se pueden apreciar las siguientes insuficiencias:

- La clasificación de los componentes en simples, medianos y complejos, no refleja con exactitud la complejidad de los datos que son gestionados. Una entrada externa con 4 elementos de datos que haga referencia a 2 archivos lógicos internos es considerada simple, mientras que otra entrada externa con 5 elementos de datos que haga referencia a 2 archivos lógicos internos es considerada como mediana.

- Límites bruscos en la asignación de la cantidad de puntos de función a un componente que resulta engañosa para el proyecto. Por ejemplo, un archivo lógico interno con 19 elementos de datos y 6 subgrupos de elementos de datos es considerado de complejidad media y le son asignados 7 puntos de función al componente; en cambio a un archivo lógico interno con 20 elementos de datos y 6 subgrupos de elementos de datos es considerado de complejidad alta y le son asignados 10 puntos de función al componente.

Para tratar los aspectos descritos anteriormente donde se presentan problemas con el manejo de informaciones incompletas, con incertidumbre o inexactas, se utilizan en la actualidad las denominadas técnicas de *soft computing*. Este término se formalizó a principios de los años 90 como una rama de la informática y se encarga de explotar la verdad parcial, la tolerancia de la imprecisión y la incertidumbre en problemas específicos. Entre las técnicas de *soft computing* se encuentra la lógica difusa, borrosa o heurística. Esta lógica se aplica a los conceptos que pueden tomar un valor indeterminado de veracidad dentro de un conjunto de valores, o sea que expresa una falta de definición en el objeto al que se aplica. [Magdalena 2010]

La lógica difusa se utiliza en procesos no lineales, con complejidad elevada y donde se manipulan definiciones y conocimientos imprecisos o subjetivos. Esta técnica proporciona una metodología simple y elegante para obtener conclusiones basadas en informaciones vagas, ambiguas, imprecisas o incompletas. Es una técnica de la inteligencia organizacional para el trabajo con informaciones con un alto grado de imprecisión [D'Negri 2006]. El aspecto sintáctico de esta técnica resulta similar a la lógica de predicados, aunque basa su semántica en la teoría de conjuntos y no permite la construcción de un sistema axiomático. [Hernández 2010]

Como consecuencia de la utilización en las diferentes técnicas de *soft computing* de un enfoque lingüístico a partir de las sucesivas investigaciones desarrolladas en este campo, surge una metodología de razonamiento, computación y toma de decisiones utilizando palabras del lenguaje natural en lugar de números, que se denominó Computación con Palabras. Con esta metodología se logra realizar un razonamiento cuando la información disponible no es precisa aprovechando la tolerancia de la imprecisión para alcanzar una mejor relación con la realidad; además de proporcionar las bases necesarias para desarrollar lenguajes de programación que pueden acercarse a los lenguajes naturales. [Zadeh 2012]

La aplicación de la Computación con Palabras para la toma de decisiones lingüísticas ha ido creciendo a través de los años al ser capaz de proporcionar herramientas cercanas a los procesos de razonamiento de los seres humanos. Pero su aplicación no se ha limitado solo a este campo, sino que también ha sido aplicada en el campo del aprendizaje, la clasificación y las bases de datos. Varios autores han propuesto diferentes modelos para llevar a cabo los procesos de esta metodología, entre estos se encuentra el modelo de representación lingüística de 2-tuplas. Este modelo como objetivo mejorar la precisión de los resultados y facilitar los procesos de la Computación con Palabras al tratar el dominio lingüístico como un dominio continuo que mantiene su base lingüística, es decir, su sintaxis y semántica. [Zulueta 2014]

Este modelo de representación de información lingüística se basa en el concepto de traslación simbólica representando la información con dos valores a los que se denomina 2-tupla lingüística. El primer valor representa a uno de los términos lingüísticos definidos y el segundo es un valor numérico que representa la traslación simbólica del término lingüístico. La traslación simbólica representa la diferencia de información que existe entre una cantidad de información y el valor entero más próximo al índice de la etiqueta lingüística más cercana. Este modelo permite realizar un proceso sencillo y sin pérdida de información donde el resultado del proceso realizado siempre se encuentra expresado en el dominio lingüístico inicial, lo que permite mejorar la precisión de sus resultados al no tener que realizar operaciones de aproximación. [Martínez 1999]

Problema

Teniendo en cuenta la situación expuesta anteriormente, se define el siguiente **problema de investigación**: ¿Cómo disminuir las imprecisiones en la estimación del esfuerzo por puntos de función para el desarrollo de software?

Objeto de investigación

La estimación del esfuerzo por puntos de función para el desarrollo de software.

Objetivo general

Definir un algoritmo para la estimación del esfuerzo a partir de puntos de función y técnicas de *soft computing* que contribuya a mejorar la precisión de las estimaciones para la gestión del alcance en el proceso de desarrollo de software.

Objetivos específicos

- ✓ Caracterizar las técnicas para la estimación del esfuerzo en proyectos de desarrollo de software y la aplicación de técnicas de *soft computing* en este contexto.
- ✓ Aplicar técnicas de *soft computing* en la estimación del esfuerzo en proyectos de desarrollo de software para mejorar la precisión de las estimaciones.
- ✓ Establecer un algoritmo para estimar el esfuerzo a partir de puntos de función y técnicas de *soft computing* en un proyecto de desarrollo de software.
- ✓ Evaluar en un caso de estudio la estimación del esfuerzo aplicando el algoritmo propuesto para validar los resultados alcanzados.

Campo de acción

La aplicación de técnicas de *soft computing* para la estimación del esfuerzo a partir de puntos de función en la gestión del alcance en el proceso de desarrollo de software.

Tipo de investigación

Teniendo en cuenta las características de la investigación, la misma se clasifica como correlacional, ya que no se necesita conocer la relación existente entre dos variables en un contexto particular. Los estudios correlacionales evalúan al grado de vinculación entre las variables lo que permite determinar el comportamiento de una variable conociendo el comportamiento de otra variable relacionada [Hernández 2010].

Hipótesis

Si se desarrolla un algoritmo para la estimación del esfuerzo a partir de puntos de función y técnicas de *soft computing*, se contribuirá a mejorar la precisión de la estimación durante la gestión del alcance del proceso de desarrollo de software. Para lo que se define como variable independiente al algoritmo para la estimación del esfuerzo a partir de puntos de función y técnicas de *soft computing* y como variable dependiente a la precisión de la estimación. En la tabla 1 y 2 se muestra la operacionalización de las variables definidas en la investigación.

Tabla 1. Análisis de la variable independiente

Variable Independiente	Dimensiones	Indicador	Unidad
Algoritmo para la estimación del esfuerzo a partir de puntos de función y técnicas de *soft computing*	Algoritmo de estimación aplicando técnicas de *sofcomputing*	Modelación a partir de conjuntos borrosos de la estimación por puntos de función	__Si __No
		Modelación a partir del modelo de representación lingüística de 2-tuplas de la estimación por puntos de función	__Si __No
	Percepción humana del tratamiento de la imprecisión	Archivos lógicos internos	__Si __No
		Archivos de interfaz externos	__Si __No
		Entradas externas	__Si __No
		Salidas externas	__Si __No
		Consultas externas	__Si __No
	Costos de desarrollo	Esfuerzo dedicado	Horas-hombre
	Aplicabilidad	Aplicación de la propuesta en un caso de estudio real	__Si __No

Tabla 2. Análisis de la variable dependiente

Variable Dependiente	Dimensiones	Indicador	Unidad
Precisión de la estimación	Tratamiento de la incertidumbre	Puntos de función sin ajustar	__Si __No
		Puntos de función ajustados	__Si __No
	Costo de implantación	Esfuerzo dedicado	Horas-hombre

Métodos de investigación

Para el desarrollo de la investigación se utilizarán los métodos teóricos y empíricos de investigación, lo que permita el análisis de todas las fuentes de información disponibles en el tema a abordar. A continuación se exponen los métodos utilizados en la investigación.

- Histórico-lógico: Para analizar los antecedentes y la evolución de los métodos de estimación utilizados en la industria del software y la utilización de técnicas de *soft computing* para mejorar sus estimaciones.
- Hipotético-deductivo: Para definir los objetivos específicos de la investigación y la hipótesis a partir del problema a resolver en el transcurso de la investigación.
- Analítico-sintético: Para analizar y sintetizar las informaciones consultadas elabora un marco teórico sobre los métodos de estimación de esfuerzo y la utilización de técnicas de *soft computing* para solucionar los problemas presentados.
- Empírico: A través de una estrategia de investigación experimental basada en tres casos de estudio para comprobar la utilidad y validez de los resultados.
- Además de la realización de encuestas y entrevistas para recopilar la información necesaria para la validación de los resultados de la investigación.

Población y muestra

El estudio de casos se realizará sobre tres proyectos desarrollados para el sistema Gestión Integral de Aduanas (GINA) que se desarrolla en el departamento de Soluciones Financieras y Aduanales del Centro para la Informatización de la Gestión de Entidades (CEIGE) de la Facultad 3 de la UCI. Los tres proyectos seleccionados fueron: Depósitos de Aduana, Despacho Comercial y Control de MTI.

Muestra

Los proyectos Depósitos de Aduana, Despacho Comercial y Control de MTI.

Diseño de experimentos

Para validar los algoritmos propuestos a través de tres casos de estudio, se utilizó el pre-experimento tipo dos: preprueba - posprueba con un solo grupo y una observación después de aplicadas las propuestas.

Experimento 1: Aplicación del método Puntos de Función tradicional, el algoritmo propuesto utilizando el modelo de representación lingüística de 2-tuplas y el algoritmo propuesto utilizando un Sistema de Inferencia Borroso en el proyecto Depósitos de Aduana.

G X O

- **Descripción de las variables**

G: Grupo de experimentación compuesto por el proyecto Depósitos de Aduana.

X: Aplicación del método Puntos de Función tradicional, el algoritmo propuesto utilizando el modelo de representación lingüística de 2-tuplas y el algoritmo propuesto utilizando un Sistema de Inferencia Borroso al grupo de experimentación definido.

O: Análisis de los resultados al aplicar los métodos de estimación en el grupo de experimentación.

Experimento 2: Aplicación del método Puntos de Función tradicional, el algoritmo propuesto utilizando el modelo de representación lingüística de 2-tuplas y el algoritmo propuesto utilizando un Sistema de Inferencia Borroso en el proyecto Despacho Comercial.

G X O

- **Descripción de las variables**

G: Grupo de experimentación compuesto por el proyecto Despacho Comercial.

X: Aplicación del método Puntos de Función tradicional, el algoritmo propuesto utilizando el modelo de representación lingüística de 2-tuplas y el algoritmo propuesto utilizando un Sistema de Inferencia Borroso al grupo de experimentación definido.

O: Análisis de los resultados al aplicar la propuesta en el grupo de experimentación.

Experimento 3: Aplicación del método Puntos de Función tradicional, el algoritmo propuesto utilizando el modelo de representación lingüística de 2-tuplas y el algoritmo propuesto utilizando un Sistema de Inferencia Borroso en el proyecto Control de MTI.

G X O

- **Descripción de las variables**

G: Grupo de experimentación compuesto por el proyecto Control de MTI.

X: Aplicación del método Puntos de Función tradicional, el algoritmo propuesto utilizando el modelo de representación lingüística de 2-tuplas y el algoritmo propuesto utilizando un Sistema de Inferencia Borroso al grupo de experimentación definido.

O: Análisis de los resultados al aplicar la propuesta en el grupo de experimentación.

Experimento 4: Análisis del impacto económico de la propuesta.

G X O

- **Descripción de las variables**

G: Grupo de experimentación compuesto por la Red de Centros de la UCI.

X: Análisis del impacto económico de la propuesta en cualquier escenario.

O: Observación del costo de la propuesta para su implantación y para su desarrollo en la herramienta GESPRO.

Experimento 5: Análisis del impacto social, lineamiento de la política económica y social del Partido y la Revolución de la propuesta.

G X O

- **Descripción de las variables**

G: Grupo de experimentación compuesto por la sociedad.

X: Análisis del impacto social, lineamiento de la política económica y social del Partido y la Revolución de la propuesta en cualquier escenario.

O: Observación después del análisis social en el grupo de experimentación.

Análisis estadístico a realizar

El análisis estadístico a realizar partirá de la observación de los resultados de la estimación en los proyectos de desarrollo de software, tomados como casos de estudio, utilizando el método Puntos de Función tradicional y los dos algoritmos propuestos en esta investigación. Para luego realizar un análisis comparativo con los resultados de estimación obtenidos en cada uno de los casos de estudio a través del método de estimación establecido en la UCI y con el tiempo real que se demoró el desarrollo de estos proyectos.

Instrumentos

Será utilizada la encuesta y la entrevista como instrumentos para para medir las variables definidas.

Novedad

La investigación desarrollada tiene como novedad la aplicación del modelo de representación lingüística de 2-tuplas de la Computación con Palabras y un Sistema de Inferencia Borroso en un algoritmo para la estimación del esfuerzo por puntos de función en la planificación del proceso de desarrollo de software.

Aporte práctico de la investigación

El aporte práctico de la investigación consiste en la mejora de la precisión de la estimación del proceso de desarrollo de un software, para realizar planificaciones más cercanas a la realidad, permitiendo obtener entonces una mayor satisfacción de los clientes.

La participación en eventos y publicaciones del autor

1. Soria Barreda, L.; Almaguer Ochoa, M.R. (2014). Aplicando lógica difusa para eliminar los límites bruscos en el método puntos de función. I Conferencia Científica Internacional Uciencia 2014.
2. Almaguer Ochoa, M.R.; Soria Barreda, L. (2013). Sistema para el despacho de buques de la aduana cubana. Ponencia en XV Convención y Feria Internacional Informática 2013. ISBN: 978-959-7213-02-4.
3. Almaguer Ochoa, M.R.; Soria Barreda, L. (2013). Optimización del acceso a base de datos del sistema GINA-APIS apoyado en la herramienta Pentaho. Ponencia en XV Convención y Feria Internacional Informática 2013. ISBN: 978-959-7213-02-4.
4. Mediana Carbó, Y.; Soria Barreda, L. (2013). Versión 1.1 del subsistema Depósitos de Aduana. VIII Peña Tecnológica 2013.

5. Almaguer Ochoa, M.R.; Soria Barreda, L. (2012). Sistema para el despacho de aeronaves de la AGR. Ponencia Duodécima Semana Tecnológica de FORDES Tecnología Convergente Presente y Futuro. ISSN: 2076-9792.
6. Almaguer Ochoa, M.R.; Soria Barreda, L. (2012). Decodificador del estándar de transferencia de información adelantada de pasajeros. Ponencia Duodécima Semana Tecnológica de FORDES Tecnología Convergente Presente y Futuro. ISSN: 2076-9792.
7. Soria Barreda, L.; Almaguer Ochoa, M.R. (2011). Aplicando una arquitectura dirigida por modelos en un proyecto de desarrollo de software. Ponencia en XIV Convención y Feria Internacional Informática 2011. ISBN: 978-959-7213-01-7.
8. Soria Barreda, L.; Almaguer Ochoa, M.R. (2011). Elevando la calidad de la ingeniería de requisitos a través del uso de ontologías. Ponencia en XIV Convención y Feria Internacional Informática 2011. ISBN: 978-959-7213-01-7.
9. Soria Barreda, L. (2010). Utilización de Ontologías en la gestión de los requisitos de un sistema de información. Segundo Taller Cubano EUREKA 2010.
10. Soria Barreda, L. (2009). Implantación de una Oficina de Proyecto. Ponencia en XIII Convención y Feria Internacional Informática. ISBN: 978-959-286-010-0.

Estructura de la tesis

En el *Capítulo 1* se elabora un marco teórico sobre los métodos de estimación de esfuerzo y la utilización de técnicas de *soft computing* para solucionar los problemas presentados en las informaciones con incertidumbre. Se abordan además los elementos necesarios para la investigación sobre lógica difusa, Sistemas de Inferencia Borroso, Computación con Palabras y el modelo de representación lingüística de 2-tuplas.

En el *Capítulo 2* se presentan los algoritmos propuestos en la investigación aplicando lógica difusa y la Computación con Palabras con el fin de obtener estimaciones del esfuerzo por puntos de función más precisas. Para esto se describen los métodos de inferencia borrosa utilizados para determinar los pesos de los componentes que permiten calcular los puntos de función sin ajustar utilizando la lógica difusa. Además de detallar las operaciones de agregación entre 2-tuplas lingüísticas al aplicar la Computación con Palabras para determinar los puntos de función sin ajustar y el factor de ajuste.

En el *Capítulo 3* se muestra la aplicación de los dos algoritmos propuestos en la investigación para estimar el esfuerzo a partir de puntos de función utilizando la lógica difusa y la Computación con Palabras, en tres proyectos de desarrollo de software seleccionados como casos de estudio. Además de comprobar la efectividad de las propuestas realizadas a través de los resultados obtenidos al comparar las estimaciones calculadas utilizando los dos algoritmos propuestos y el método tradicional, respecto al esfuerzo real que fue necesario dedicar en los tres casos de estudio.

Para concluir este trabajo se exponen las conclusiones y recomendaciones de la investigación realizada, así como las bibliografías que fueron consultadas para su elaboración.

CAPÍTULO 1: ANÁLISIS DE LOS MÉTODOS DE ESTIMACIÓN DEL ESFUERZO

Los estudios que han sido realizados en la actualidad para determinar los factores asociados al éxito de un proyecto de desarrollo de software, han señalado la importancia que tiene la correcta definición y gestión del mismo. Para lograr esto resulta indispensable realizar estimaciones cada vez más precisas sobre el esfuerzo que será necesario dedicar al desarrollo del software. La estimación constituye una de las tareas más difíciles a realizar en la gestión de un proyecto de desarrollo de software, donde a pesar de haberse desarrollado numerosas técnicas, no existe un consenso sobre cuál utilizar en cada situación debido a las dificultades que todavía persisten y que afectan la exactitud de sus estimaciones.

En este capítulo se realiza una caracterización de los métodos para la estimación del esfuerzo en proyectos de desarrollo de software. Se enfatiza en los métodos Puntos de Función, COSMIC y Puntos de Casos de Usos al ser los métodos más referenciados para calcular el tamaño de un software. Además de revisarse la aplicación de técnicas de *soft computing* para mejorar la precisión de las estimaciones de los métodos analizados.

1.1. Análisis bibliométrico

Para validar la actualidad de la bibliografía consultada, así como para realizar una caracterización de la misma, se realizó un análisis bibliométrico. En la tabla 3 se muestra el resumen de las fuentes consultadas, destacando aquellas de los últimos cinco años.

Tabla 3: Resumen bibliométrico

Tipo de bibliografía	**Últimos 5 años**		**Años anteriores**	
	Cantidad	**%**	**Cantidad**	**%**
Libros	2	4.55	2	4.55
Tesis de doctorados	1	2.27	5	11.36
Artículos en revistas	9	20.46	7	15.91
Memorias de eventos	6	13.63	6	13.63
Informes	4	9.09	0	0
Normas y Resoluciones	2	4.55	0	0
Total	**24**	**54.55**	**20**	**45.45**

A raíz del estudio anterior se puede concluir que el 55 % de la bibliografía consultada corresponde a los últimos cinco años; de estas se encuentran en idioma inglés veintidós, lo que representa el 50 % del total consultado. Se debe señalar que las seis tesis de doctorados consultadas fueron elaboradas en los años 1999, 2000, 2005, 2006, 2007 y 2014. También se debe destacar el predominio de las publicaciones en el continente americano con un 61 %, con un 34 % de bibliografía consultada perteneciente a los Estados Unidos. Además los tres años en que se concentra la mayor cantidad de bibliografías consultadas de esta investigación está dentro de los últimos 5 años; siendo el 2010 el

año más representado con 9 publicaciones lo que significa un 20 % del total de la bibliografía consultada, así mismo le siguen el año 2011 y 2014 con 6 publicaciones, lo que representa un 14 % en ambos años.

Para la obtención de la documentación se realizaron búsquedas con la herramienta de búsqueda en internet Google Scholar, así como en las bases de datos de publicaciones referenciadas Scielo, Redalyc y Elsevier, a través de la biblioteca de la UCI. La documentación consultada demuestra la creciente utilización de las técnicas de *soft computing* para mejorar la precisión de las estimaciones del tamaño de software en la industria de desarrollo de software. Sin embargo, todavía persisten dificultades para estimar con precisión el tamaño de un software. Además de no haber sido utilizado el modelo de representación lingüística de 2-tuplas perteneciente a la Computación con Palabras para mejorar la precisión de las estimaciones.

1.2. Estimación de software

Realizar una estimación de software lo más cercana a la realidad resulta de vital importancia en cualquier proyecto de desarrollo de software para su planificación, así como para determinar los esfuerzos y costos requeridos para su construcción. En la industria de desarrollo de software se utilizan en la actualidad una gran variedad técnicas para estimar; estas técnicas pueden ser agrupadas en tres categorías fundamentalmente. Estas categorías son:

Juicio de expertos: estas técnicas utilizan los criterios de las personas que son consideradas expertos en el proyecto. Estos criterios vienen fundamentados con los datos históricos de proyectos anteriores y las estimaciones realizadas en proyectos con características similares. Este método tiene el inconveniente de ser muy subjetivo y por tanto no poderse reutilizar; además de no poder contar con una argumentación analítica.

Modelos algorítmicos: constituye la categoría más popular en la industria y la literatura, ejemplo de estos modelos son COCOMO, SLIM y SEER-SEM. La popularidad de estos modelos se basa en la argumentación analítica con que cuentan, lo que permite su utilización y adaptación a cualquier entorno. Para realizar la estimación estos modelos se centran fundamentalmente en el tamaño del software.

Aprendizaje automático: esta categoría la conforman técnicas definidas recientemente en la industria donde se conjuga la inteligencia artificial y los modelos algorítmicos. Dentro de estas técnicas se pueden encontrar redes neuronales, lógica difusa, árboles de regresión y algoritmos genéticos.

Se debe señalar que ninguna de estas técnicas es perfecta, es decir, que sean capaces de proporcionar un valor que se ajuste a la realidad y que además encajen en todas las situaciones que se presentan a diario en la industria de desarrollo de software. Es por esta situación, que se sigue investigando en esta área con el objetivo de mejorar las estimaciones que se realicen, lo que permita a los proyectos ajustarse a los costos y fechas planificadas. Para esto, se considera que un factor

importante a tener en cuenta es el tamaño del software, ya que la mayoría de las técnicas utilizadas parte a partir de este valor, por lo que mientras más ajustado se encuentre este valor, más certera será la estimación de software que se lleve a cabo.

1.3. Estimación del tamaño del software

Un factor importante relacionado al éxito de los proyectos de desarrollo de software y por consiguiente a la calidad de sus productos entregables, lo constituye la determinación del tamaño del software que se desarrolla, lo cual permitirá determinar el esfuerzo necesario para su construcción. Este valor en la actualidad es utilizado en la gestión y control del proceso de desarrollo de software, por lo que su utilidad es mucho mayor si se obtiene en etapas tempranas del desarrollo.

El tamaño de un software puede ser determinado utilizando diferentes técnicas. En la actualidad existen técnicas que son utilizadas luego de construido el software (las líneas de código) y otras que son utilizadas antes de ser construido, entre las que se encuentran los métodos de medición del tamaño funcional al cuantificar los requisitos funcionales pactados. Los métodos existentes para medir el tamaño de un software pueden ser clasificados en uno de los dos siguientes enfoques. [Condori 2007]

Enfoque técnico: se tienen en cuenta las características técnicas del software, por lo que son mediciones que se realizan a posteriori.

Enfoque funcional: se tienen en cuenta la funcionalidad requerida por el usuario, por lo que el software puede ser medido en etapas tempranas e independientemente de la tecnología utilizada.

Esta investigación se centra en las técnicas aplicables durante el ciclo de desarrollo del software, debido a la importancia que conlleva lograr una estimación mucho más precisa que permita una correcta planificación del proceso. A continuación se realiza un estudio de las diferentes técnicas que existen para determinar el tamaño de un software durante su construcción.

1.3.1. Método Puntos de Función

El método Puntos de Función fue definido en el año 1979 por Allan Albrecht basado en el estudio de 22 proyectos de la empresa IBM. Este método tiene como objetivo medir el tamaño de una funcionalidad de software entregada al usuario independientemente del lenguaje y la tecnología utilizada para su desarrollo [Albrecht 1979]. A través de los años este método ha sido actualizado (1983, 2000 y 2003), con el objetivo de obtener estimaciones cada vez más cercanas a la realidad y teniendo como base el diseño lógico del software por encima de la cantidad de líneas de código.

En su primera versión se propuso clasificar los componentes del software en entradas, salidas, consultas y archivos. En su siguiente versión los archivos se dividieron en archivos lógicos internos y archivos de interfaz externa. A cada uno de estos componentes se les asignaba un puntaje teniendo en cuenta su tamaño y complejidad. Este puntaje propuesto se muestra en las tablas 4 y 5.

Tabla 4. Pesos en el método de puntos de función en su versión de 1979 [Albrecht 1979]

Tipos de función	Peso
Archivos	10
Entradas	4
Salidas	5
Consultas	4

Tabla 5. Pesos en el método de puntos de función en su versión de 1983 [Dolado 1999]

Tipos de función	Peso		
	Complejidad Baja	Complejidad Media	Complejidad Alta
Archivos lógicos internos	7	10	15
Archivos de interfaz externos	5	7	10
Entradas externas	3	4	6
Salidas externas	4	5	7
Consultas externas	3	4	6

Luego de establecer los pesos a cada componentes se calculan los llamados puntos de función sin ajustar o UFP, por sus siglas en inglés, que consistían en la sumatoria de los pesos de cada componente. Luego este valor se ajustaba teniendo en cuenta las características generales del software y se obtenían los llamados puntos de función ajustados.

Este método es desarrollado en la actualidad por The International Function Point Users Group (IFPUG); organización internacional sin fines de lucro, que es gobernada por sus miembros y promueve la gestión del software a través de la utilización de esta y otras técnicas. A partir de este momento, se propone medir la funcionalidad que el software es capaz de proporcionar a los usuarios, teniendo como base su diseño lógico. En cada una de las nuevas versiones de este método, la IFPUG, proporciona un manual de prácticas de conteo donde se realizan todas aclaraciones necesarias para la exitosa utilización del método. [Ifpug 2010]

En las nuevas versiones del método se le definen un conjunto de reglas para determinar la complejidad de cada uno de los componentes. En el caso de los archivos, tanto los lógicos internos como los de interfaz externa, la complejidad se basa en la cantidad de elementos de datos y tipos de registros en estos archivos. En el resto de los componentes, dígase entradas, salidas y consultas externas, se basa en la cantidad de elementos de datos y tipos de archivos que se referencian.

Además de incorporarle catorce factores técnicos y de calidad para realizar el ajuste de los puntos de función. En las siguientes versiones de este método, la IFPUG, incorpora en su manual de prácticas de conteo, aclaraciones a las reglas, guías y criterios, pero siempre se mantiene fiel a su estructura. En esencia este método consiste en contar en el software las funciones de datos y transaccionales. A continuación se definen los conceptos utilizados en este método. [Ifpug 2010]

Funciones de datos: lo constituye las funcionalidades que satisfacen los requerimientos de datos, ya sea internos como externos. Se clasifica en archivos internos lógicos y archivos de interfaz externos.

Archivos internos lógicos (AIL): lo constituyen los grupos de datos que se relacionan lógicamente, así como las informaciones identificables por los usuarios de control que se encuentran dentro de los límites de la aplicación a medir.

Archivos de interfaz externos (AIE): lo constituyen los grupos de datos que se relacionan lógicamente, así como las informaciones identificables por los usuarios de control que no se encuentran dentro de los límites de la aplicación a medir.

Funciones transaccionales: lo constituyen las funcionalidades que se encargan de procesar los datos. Se clasifica en entradas externas, salidas externas y consultas externas.

Entrada externa (EE): lo constituye las funcionalidades que procesan los datos o las informaciones de control identificables por los usuarios provenientes desde fuera de los límites de la aplicación con el propósito de mantener los AIL y modificar el comportamiento del sistema.

Salida externa (SE): lo constituye las funcionalidades que envían los datos o las informaciones de control identificables por los usuarios hacia fuera de los límites de la aplicación, con el objetivo de presentarle al usuario la información haciendo uso de una lógica de procesamiento, que contiene al menos un cálculo matemático, derivación de datos, mantenimiento de los AIL o la modificación del comportamiento del sistema.

Consulta externa (CE): lo constituye las funcionalidades que envían los datos o las informaciones de control identificables por los usuarios hacia fuera de los límites de la aplicación, con el objetivo de presentarle al usuario la información recuperada de un AIL o AIE.

Para cada AIL y AIE se definen los siguientes registros compuestos por campos.

Tipo de elementos de datos (TED): representa un campo único que no se repite y que es reconocido por el usuario.

Tipo de elementos de registros (TER): representa a un subgrupo de elementos de datos que son reconocidos por el usuario.

Para cada EE, SE y CE contiene los siguientes tipos de referencias a los campos en los archivos.

Tipo de archivo referenciado (TAR): lo constituye un AIL o un AIE leído y mantenido en una función transaccional.

Las cantidades de TED y TER determinan la cantidad de puntos de función del AIL o AIE. Estas cantidades se muestran en las tablas 6 y 7. Las cantidades de TAR y TED determinan la cantidad de puntos de función de las EE, SE y CE. Estas cantidades se muestran las tablas 8, 9 y 10.

Tabla 6. Puntos de función para los AIL teniendo en cuenta los elementos TED y TER [Ifpug 2010]

Cantidades	1 a 19 TED	20 a 50 TED	51 o más TED
1 TER	7	7	10
2 a 5 TER	7	10	15
6 o más TER	10	15	15

Tabla 7. Puntos de función para los AIE teniendo en cuenta los elementos TED y TER [Ifpug 2010]

Cantidades	1 a 19 TED	20 a 50 TED	51 o más TED
1 TER	5	5	7
2 a 5 TER	5	7	10
6 o más TER	7	10	10

Tabla 8. Puntos de función para las EE teniendo en cuenta los elementos TED y TAR [Ifpug 2010]

Cantidades	1 a 4 TED	5 a 15 TED	16 o más TED
0 o 1 TAR	3	3	4
2 TAR	3	4	6
3 o más TAR	4	6	6

Tabla 9. Puntos de función para las SE teniendo en cuenta los elementos TED y TAR [Ifpug 2010]

Cantidades	1 a 5 TED	6 a 19 TED	20 o más TED
0 o 1 TAR	4	4	5
2 a 3 TAR	4	5	7
4 o más TAR	5	7	7

Tabla 10. Puntos de función para las CE teniendo en cuenta los elementos TED y TAR [Ifpug 2010]

Cantidades	1 a 5 TED	6 a 19 TED	20 o más TED
0 o 1 TAR	3	3	4
2 a 3 TAR	3	4	6
4 o más TAR	4	6	6

Insuficiencias presentadas por el método Puntos de Función

A pesar de la popularidad alcanzada por el método Puntos de Función para la estimación de un software, este método todavía presenta insuficiencias en la clasificación de los componentes del software a medir (simples, medianos y complejos) al tener en cuenta la complejidad de los datos que se manejan. Por ejemplo, una entrada externa con 4 elementos de datos que haga referencia a 2 archivos lógicos internos es considerada simple, mientras que otra entrada externa con 5 elementos de datos que haga referencia a 2 archivos lógicos internos es considerada como mediana.

Además, la cantidad de puntos que se le propone asignar a un componente teniendo en cuenta los rangos definidos en el método para su clasificación, provoca la presencia de los llamados límites

bruscos en casos donde el esfuerzo que se le debe dedicar es muy similar, lo que aleja las estimaciones realizadas de la realidad. Por ejemplo, un archivo lógico interno con 19 elementos de datos y 6 subgrupos de elementos de datos es considerado de complejidad media y le son asignados 7 puntos de función al componente; en cambio a un archivo lógico interno con 20 elementos de datos y 6 subgrupos de elementos de datos es considerado de complejidad alta y le son asignados 10 puntos de función al componente.

A todo esto se le debe sumar que en una organización donde no se utilice este método frecuentemente en la estimación de sus proyectos de desarrollo de software, le resultará muy complejo la formación de su personal para asegurar que los criterios que se manejen en el recuento sean homogéneos. Por lo que se requerirá una dedicación adicional en los proyectos de desarrollo de software y un fuerte compromiso de la dirección de la organización. También se considera que resulta muy costosa la utilización de este método en las organizaciones que mantienen software que han sido adquiridos a terceros. Además de considerarse en algunas literaturas que el factor de ajuste que se calcula teniendo en cuenta las características generales del sistema es de una utilidad dudosa para la estimación.

1.3.2. Método COSMIC

El método COSMIC-FFP fue publicado en 1999 por el Consorcio Internacional de Medición de Software Común (COSMIC) y constituye un método de medición del tamaño de las funcionalidades de un software de segunda generación. Este método tiene como propósito estandarizar un método que permita medir el tamaño funcional de las aplicaciones de gestión y las aplicaciones de tiempo real. Para su definición fueron recolectados por varios años los datos de diferentes proyectos de desarrollo de software de forma tal que le permitiera establecer patrones de comportamiento con estos datos.

Este método tiene sus bases en el método propuesto por St-Pierre en 1997 para medir el tamaño funcional de las aplicaciones de tiempo real, pero también puede ser utilizado en la medición de otros tipos de aplicaciones. Este método consiste en la aplicación de un conjunto de procedimientos y reglas a un componente del software de forma tal que permita convertirlo en un modelo al que se le aplicarán otro conjunto de procedimientos y reglas para medirlo. Para lograr esto se definen en este modelo las siguientes dos fases.

Fase de mapeo: proporciona un modelo de software genérico teniendo como entrada los requerimientos funcionales y aplicando un conjunto de reglas y procedimientos.

Fase de medición: proporciona un valor directamente proporcional al tamaño funcional del software teniendo como entrada un modelo de software genérico.

El modelo genérico que se utiliza en este método considera que el software contiene una frontera en la que existen procesos funcionales y almacenamiento persistente, además del movimiento propio de

los grupos de datos a través de su frontera y el almacenamiento persistente. Por cada movimiento de datos que se realice en la aplicación se mide una unidad de tamaño funcional de COSMIC, llamada CFSU, por sus siglas en inglés. A continuación se definen estos conceptos que utiliza el método COSMIC.

Proceso funcional: responde a un conjunto de requerimientos del usuario para el movimiento de datos, ejecutado a partir de un evento disparador por medio de un actor y que se completa al dar una respuesta de todo lo realizado al evento que lo disparó.

Grupos de datos: lo constituye un conjunto de atributos de datos único, no ordenado, vacío ni redundante que describen un aspecto complementario de un mismo objeto y que puede persistir en el tiempo.

Entrada: lo constituye el tipo de movimiento de datos desde el usuario hasta un proceso funcional a través de los límites de la aplicación.

Salida: lo constituye el tipo de movimiento de datos desde un proceso funcional hasta el usuario a través de los límites de la aplicación.

Lectura: lo constituye el tipo de movimiento de datos desde el almacenamiento que lo persiste hasta el proceso funcional que lo solicita.

Escritura: lo constituye el tipo de movimiento de datos desde el proceso funcional hasta el almacenamiento persistente.

Este método define el tamaño de un componente de un software como la suma de todos los tamaños calculados para todos los procesos funcionales definidos. El tamaño de un proceso funcional se determina a su vez, como la sumatoria del tamaño de todos los movimientos de datos que se realicen en este proceso funcional. [Machado 2006]

Insuficiencias presentadas por el método COSMIC

Comparado con otros métodos de medición, el método COSMIC requiere un mayor tiempo de preparación por parte del personal que medirá el software debido a la documentación que se propone registrar para la definición de la medición a realizar. A pesar de haber sido aceptado como estándar internacional en la norma ISO/IEC 19761 en el año 2002, este método no ha logrado una gran popularidad en la industria de desarrollo de software, lo que le ha privado de adquirir una mayor maduración.

Por la poca madurez que presenta este método entonces, no se puede determinar si su propuesta de no tener en cuenta la cantidad de atributos que se manejan en un movimiento de datos puede ser considerado una debilidad o fortaleza. De igual forma, el método no propone una clasificación de las funcionalidades para asignarles según su complejidad y cantidad un peso que determine la cantidad

de puntos de función, sino que se le pueden atribuir una cantidad ilimitada de puntos de función según el recuento que se realice al proceso funcional.

1.3.3. Método Puntos de Casos de Uso

El método Puntos de Casos de Uso para la estimación del tamaño de software fue creado en el año 1993 por Gustav Karner en su tesis de maestría, que estuvo dirigida por Ivar Jacobson quien fuera el creador de los casos de uso. Este método fue creado teniendo como base el método Puntos de Función y ha sido utilizado por la empresa Rational Software, luego IBM, a través de los años con buenos resultados. Este método tiene como base la utilización en el proceso de desarrollo del software una metodología orientada a objetos y contiene dos etapas. Una primera etapa donde se determinan los puntos de casos de uso de sin ajustar y una segunda etapa donde se ajustan los puntos de casos de usos teniendo en cuenta factores técnicos y de ambiente. [Remón 2011]

Los puntos de casos de uso sin ajustar se encuentran determinados por la sumatoria de los factores de peso de los actores y casos de usos sin ajustar. Para determinar el factor de peso de los actores, estos deberán ser clasificados según su complejidad, lo que responde a que si se trata de una persona o sistema, así como la forma en que el actor interactúa con el sistema. En la tabla 11 se muestran los factores de peso propuestos en este método para cada tipo de actor.

Tabla 11. Factores de peso de los actores en el método Puntos de Casos de Uso [Nassif 2011 b]

Tipo de actor	Descripción	Factor
Simple	Interacción entre sistemas a través de interfaces de programación.	1
Promedio	Interacción entre sistemas a través de protocolos o interfaces basada en textos.	2
Complejo	Persona que interactúa con el sistema a través de una interfaz gráfica.	3

Para determinar el factor de peso en los casos de uso, estos deberán ser clasificados según su complejidad, lo que responde a la cantidad de transacciones que se realizan en el mismo. En la tabla 12 se muestran los factores de peso propuestos en este método para las transacciones.

Tabla 12. Factores de peso de los actores en el método Puntos de Casos de Uso [Nassif 2011 b]

Tipo de caso de uso	Descripción	Factor
Simple	De 1 a 3 transacciones	5
Promedio	De 4 a 7 transacciones	10
Complejo	Mayor a 7 transacciones	15

Luego de calcular los puntos de casos de uso sin ajustar, se debe proceder a su ajuste, lo que incluye la consideración de factores técnicos y ambientales en el proceso de desarrollo. Los factores técnicos se encuentran relacionados a la complejidad del proyecto, mientras que los factores ambientales se orientan hacia la eficiencia y la productividad del equipo de desarrollo. En las tablas 13 y 14 se muestran los factores técnicos y ambientales propuestos por este modelo respectivamente.

Tabla 13. Factores técnicos propuestos en el método Puntos de Casos de Uso [Nassif 2011 b]

Número	Factor	Peso
1.	Sistemas distribuidos	2
2.	Desempeño de la aplicación	1
3.	Eficiencia para usuario final	1
4.	Procesamiento interno complejo	1
5.	Reusabilidad	1
6.	Instalación sencilla	0.5
7.	Usabilidad	0.5
8.	Portabilidad	2
9.	Adaptabilidad	1
10.	Concurrencia	1
11.	Características especiales de seguridad	1
12.	Acceso directo para terceros	1
13.	Facilidades especiales en el entrenamiento de usuarios	1

Tabla 14. Factores ambientales propuestos en el método Puntos de Casos de Uso [Nassif 2011 b]

Número	Factor	Peso
1.	Familiaridad con los objetos	1.5
2.	Requisitos estables	2
3.	Capacidad de análisis	0.5
4.	Experiencia con la aplicación	0.5
5.	Experiencia con el trabajo orientado a objetos	1
6.	Motivación	1
7.	Dificultades con el lenguaje de programación	-1
8.	Trabajadores a tiempo parcial	-1

Para ajustar los puntos de casos de uso se calcula un único factor técnico y un único factor ambiental, que se encuentran determinados por las fórmulas 1 y 2 respectivamente según [Nassif 2011 b]. Para el factor técnico se define que C1 = 0.6 y C2 = 0.01, mientras que para el factor ambiental se define que C1 = 1.4 y C2 = -0.03. En ambos factores la variable F_i representa un valor entre 0 y 5, donde 0 significa que el factor no se considera relevante, el 5 representa que es esencial para el proceso y el valor 3 significa que no es esencial pero tampoco irrelevante, por lo que es considerado promedio. Además, en ambos factores la variable W_i representa el peso que se le asocia al factor para su ponderación.

$$TF = C1 + C2 \sum_{i=1}^{13} F_i * W_i. \quad (1)$$

$$EF = C1 + C2 \sum_{i=1}^{8} F_i * W_i. \quad (2)$$

Finalmente, el ajuste de los puntos de casos de uso resulta en la multiplicación de los tres valores que han sido descritos en esta sección: los puntos de casos de uso sin ajustar, el factor técnico y el factor ambiental. Con esta cantidad de puntos de casos de uso ajustados que ha sido calculado, se

procede entonces al cálculo del esfuerzo que demandará el desarrollo del sistema medido. Para esto en el método se propuso inicialmente calcularlo mediante la multiplicación de la cantidad de puntos de casos de uso y el valor 20 horas-personas. Este valor sugerido para calcular el esfuerzo de desarrollo ha sido mejorado a través del tiempo.

Schneider y Winter, en la segunda edición de su guía práctica para aplicar casos de usos en el año 2001, incorporan los factores ambientales en el cálculo del esfuerzo, donde se propone utilizar un valor de horas-personas diferente dependiendo de estos factores. La propuesta consiste en contar los factores que cumplen la siguiente condición, para los primeros 6 factores que el valor otorgado se encuentre por debajo de 3 y para los últimos dos factores que el valor otorgado se encuentre por encima de 3. Para los proyectos donde se cuenten con uno o dos factores que cumplan esta condición se utilizará el valor de 20 horas-personas. En los casos que se cuenten tres o cuatro factores que cumplan esta condición se utilizará el valor de 28 horas-personas. Así como en los casos que se cuenten cinco o más factores que cumplan esta condición, se sugiere la reestructuración del equipo de desarrollo ya que con el equipo propuesto se corren grandes riesgos de fracasar, en los casos en los que se decida no reestructurar el equipo de desarrollo se propone utilizar el valor de 36 horas-personas. [Schneider 2001]

Insuficiencias presentadas por el método Puntos de Casos de Uso

Al método Puntos de Casos de Uso se le ha cuestionado el peso asignado a los actores y la influencia que puedan tener estos en la estimación cuando ya se tiene en cuenta en cada uno de los casos de uso su actuar; así como la diferenciación que realiza en los factores propuestos a la hora de determinar los factores técnicos y ambientales. Por su parte los factores que se utilizan para calcular el costo del proyecto han sido objetos de crítica por la incertidumbre que rodea a esta información.

Este método también presenta problemas a la hora de asignar la cantidad de puntos de casos de uso teniendo en cuenta los rangos definidos para la clasificación del caso del uso a medir. Esta propuesta de asignación provoca la presencia de los llamados límites bruscos en casos donde el esfuerzo a dedicar es muy similar, lo que provoca el distanciamiento de las estimaciones de la realidad. Por ejemplo, un caso de uso donde se realizan 7 transacciones es considerado de complejidad media y le son asignados 10 puntos de casos de uso, en cambio un caso de uso donde se realizan 8 transacciones es considerado de complejidad alta y le son asignados 15 puntos de casos de uso.

Se debe señalar que para la utilización de este método de estimación, el proyecto de desarrollo de software debe utilizar para la captura de los requisitos funcionales del sistema la técnica de casos de uso. Esta técnica ha sido ampliamente utilizada desde su creación, pero en los últimos años el número de organizaciones que utilizan el modelado por procesos para describir el negocio del sistema a desarrollar ha ido en aumento, lo que va en detrimento de esta técnica. Además, la inexistencia de un estándar para describir los casos de uso que sean definidos es un factor que dificulta la aplicación de este método al atentar contra la homogeneidad del conteo.

1.4. Utilización de técnicas de *soft computing* en la estimación del tamaño de software

El término *soft computing* se comenzó a utilizar en la rama de la informática en la década del 90 y abarca las técnicas utilizadas para dar solución a los problemas de información con incertidumbre, inexacta e incompleta. Estas técnicas intentan complementarse entre ellas mismas y tienen mayor similitud con los procesos matemáticos que con las técnicas biológicas tradicionales. Los principales componentes de *soft computing* son la lógica difusa, las redes neuronales, el razonamiento probabilístico y la computación evolutiva. [Magdalena 2010]

Las técnicas de *soft computing* han sido utilizadas en los últimos años para tratar de solucionar los problemas asociados al manejo de informaciones incompletas, con incertidumbre o inexactas, en los procesos de estimación del tamaño de software que se llevan a cabo con los diferentes métodos de estimación que se utilizan en la industria de desarrollo de software. A continuación se realiza un análisis de las diferentes soluciones que se le han propuesto a los métodos de estimación Puntos de Función y Puntos de Casos de Uso para mejorar sus resultados aplicando las técnicas de *soft computing* que existen.

1.4.1. Puntos de Casos de Uso

En la 16 Conferencia Internacional de la IEEE sobre herramientas con Inteligencia Artificial en el año 2004, Braz y Vergilio propusieron una extensión del método Puntos de Casos de Uso utilizando la teoría de conjuntos difusos, a la cual se bautizó como FUSP, por sus siglas en inglés, Fuzzy Use case Size Points. Esta propuesta fue validada con los datos registrados en los proyectos de una compañía privada. En la evaluación de la propuesta se aprecian mejores resultados que el método tradicional en la mayoría de los módulos, pero también se detectan resultados peores en algunos módulos debido a la incertidumbre subyacente en las primeras fases del proyecto. [Braz 2004]

En la Conferencia Internacional de Inteligencia Computacional e Ingeniería de Software realizada en el año 2009, fue propuesta una extensión al método Puntos de Casos de Uso utilizando la teoría de conjuntos difusos y las redes bayesianas. Para lograr esto se propuso extender a cinco las categorías propuestas de los casos de uso para su clasificación. En esta propuesta se construye un modelo probabilístico de costo al integrar la teoría de conjuntos difusos con las redes bayesianas, lo que permite proporcionar una distribución probabilística del costo y una clasificación gradual refinada para mitigar la incertidumbre de los factores utilizados. Esta propuesta fue validada a través de la comparación de los resultados de estimación obtenidos en dos casos de estudios. [Fan 2009]

En [Nassif 2010] se propone la clasificación de los casos de uso en 10 categorías borrosas de acuerdo a la cantidad de transacciones, para luego ser calculado el tamaño del software con una red neuronal. Para las entradas de la red neuronal se definieron 10 vectores que modelaban la complejidad de los casos de uso y 3 vectores para representar la complejidad de los actores. Con el enfoque de la lógica difusa se planteó realizar una asignación gradual de los pesos a los casos de uso según su complejidad. La red neuronal fue utilizada como una caja negra para mapear los

vectores de entrada con el tamaño del software. Este método demostró una mejora en un 22 % respecto al método tradicional propuesto por Karner teniendo en cuenta los resultados que se obtuvieron al aplicarse en 20 proyectos diferentes.

Para determinar la complejidad del actor en el método y el peso asociado al mismo en [Nunes 2011] se propone una nueva métrica basada en la heurística de los conceptos asociados a la interacción entre la computadora y las personas. En esta nueva propuesta de clasificación se definen seis categorías donde se separan los actores humanos y los actores del sistema, así como se amplía hasta 5 el valor del peso asignado. De esta forma se propone asignar un peso de 1 a los actores del sistema con complejidad simple, un peso de 2 a los actores del sistema con complejidad media, un peso de 3 a los actores humanos con complejidad simple y a los actores del sistema con alta complejidad, un peso de 4 a los actores humanos con complejidad media y un peso de 5 a los actores humanos con una alta complejidad.

También se propuso mejorar el conteo de los puntos de casos de uso mediante la utilización de la lógica difusa en [Lee 2011]. Para esto se propone la aplicación de una nueva métrica para determinar la complejidad de los casos de uso basándose en su relación con los objetivos que impulsa a través de funciones de pertenencia. La teoría de conjuntos difusos fue utilizada para definir los valores de satisfacción de los casos de usos respecto a los objetivos definidos, para luego aplicar un conjunto de reglas difusas que permitieran calcular los pesos asociados. El método utilizado para realizar el proceso de concreción de los resultados obtenidos al aplicar las reglas fue el centro de gravedad. Los autores de esta propuesta se centraron en demostrar la viabilidad del método más que en la comparación de los resultados con otros métodos, realizando la validación del mismo a través de un caso de estudio.

En ese mismo año fue propuesta también la aplicación de un modelo de regresión para determinar la relación no lineal que existe entre el esfuerzo y el tamaño en [Nassif 2011 a], normalizando los datos a través de transformaciones logarítmicas. Para ajustar el factor de productividad en el modelo de regresión propuesto, el colectivo de autores propuso en [Nassif 2011 b] adicionar un sistema de inferencia borrosa del tipo Sugeno. Esta propuesta fue evaluada en 24 proyectos diferentes, donde se obtuvo una mejora de un 11% respecto al método tradicional propuesto por Karner y de un 7% respecto al modelo que fuera propuesto por [Schneider 2001].

Entre las últimas investigaciones realizadas para mejorar las estimaciones utilizando el método Puntos de Casos de Uso se encuentra la propuesta de [Nassif 2014] para calibrar la complejidad de los pesos asignados. En este artículo publicado se propone realizar el cálculo de los puntos de casos de uso a partir del diagrama de casos de uso y los escenarios que sean descritos en el modelo definido. Para realizar estos cálculos se propuso la utilización de una red neuronal y la lógica difusa, lo cual, a criterio de los autores, permitirá que la cantidad de puntos de casos de uso obtenida se encuentre calibrada.

1.4.2. Puntos de Función

Los primeros pasos estuvieron encaminados a modelar probabilísticamente el comportamiento del método en [Yau 1998] para proponer los valores de las 14 características del sistema que se utilizan para ajustar los puntos de función. En [De Souza 2003] se propone por primera vez la utilización de los conjuntos difusos para mejorar las estimaciones de este método con la utilización de un sistema que permite calcular los pesos de los componentes utilizando el modelo propuesto; pero en este trabajo se debe señalar que para la definición del modelo se utilizó una pequeña base de datos compuesta por sistemas heredados desarrollados en Visual Basic y Microsoft Acces principalmente, lo que compromete la fiabilidad del modelo creado.

Siguiendo esta tendencia, en [Xia 2008] se propone la utilización de un enfoque neuro-fuzzy, la combinación de lógica difusa con redes neuronales, para calibrar los valores de los pesos del método utilizando los datos disponibles en el repositorio de proyectos de software del Grupo Internacional de Estándares de Medición de Software (ISBSG, por sus siglas en inglés). Este trabajo permite reflejar en el método las tendencias de la industria en el desarrollo de software al aprender de los datos históricos estudiados en este repositorio, lo que puede distorsionar los resultados al aplicarse en un entorno con características propias para el desarrollo de software. Además de mantener las tres categorías para clasificar la variable relacionada con los elementos de datos, lo que implica continuar estimando con intervalos grandes.

Para mejorar la estimación del método Puntos de Función, también se ha propuesto la combinación de la lógica difusa y los métodos de interpolación en [Chen 2010], pero en dicha investigación los autores reconocen que los parámetros de la función de membresía y los nodos de interpolación deben ser mejorados debido a que la cantidad de datos históricos utilizados en el estudio fue baja. Además según en el estudio realizado por [Ferreira 2014] sobre la utilización de técnicas de inteligencia artificial para la estimación del esfuerzo en proyectos de software, se puede concluir que la tendencia en los autores gira en torno a proponer nuevos modelos basados en la utilización de los datos históricos de estimaciones de proyectos anteriores, a los que se les aplica diferentes técnicas de extracción del conocimiento para realizar la estimación del nuevo proyecto a desarrollar.

Según el análisis de los trabajos anteriores y el estudio realizado por [Ferreira, 2014], los modelos propuestos hasta el momento, que cuentan con una validación exitosa al utilizar una técnica de *soft computing* para mejorar las estimaciones utilizando el método Puntos de Función, han utilizado los registros históricos de estimaciones de proyectos anteriores para realizar los cálculos de los valores que se proponen. Esta situación implica que estas propuestas no puedan ser utilizadas en las empresas de desarrollo de software que no cuenten con los suficientes datos históricos sobre las estimaciones realizadas en proyectos ya terminados que le permitan realizar un aprendizaje de sus características propias; ya que estas características pueden diferir en gran medida con las tendencias de la industria y los datos en las que se basan estas propuestas se encuentran acorde a dichas tendencias.

1.5. Lógica difusa y Sistemas de Inferencia Borrosos

La lógica borrosa se desarrolló a partir de la teoría básica de los conjuntos borrosos establecidos por Lofti Zadeh en 1965 en la Universidad de Berkeley, donde cada conjunto borroso es un conjunto de elementos cuya pertinencia al mismo es gradual y no absoluta. Con esto se desarrolló una lógica que no solo contemplaba las opciones falsas y verdaderas, sino también un conjunto de variables respuestas que se encuentran entre estas. Se puede catalogar como una alternativa a la lógica discreta donde se descubren diversos grados de pertenencia; se debe destacar que la lógica difusa en ningún momento rechaza la lógica discreta. [Ballester 2006]

Para poder aplicar la teoría de la lógica difusa en la solución de un problema, se deben definir las entradas, salidas y reglas difusas a aplicar. Para esto se le define al conjunto de valores de entrada y salida, una función de pertenencia μ(x), la cual define el grado de pertinencia de cada uno de los elementos al conjunto al mapearlos con un valor entre 0 y 1; por lo que se puede encontrar a un mismo elemento en varios de los conjuntos borrosos definidos, donde en cada uno de estos conjuntos tendrá un cierto grado de pertenencia. En la actualidad existen diferentes tipos de función de pertinencia, entre estos se pueden encontrar los tipos triangular, trapezoidal, gaussiano, lineal y singleton. Un conjunto borroso A se define mediante notación matemática como $A = \{(x, \mu_A(x)) / x \in U\}$, así mismo su función de pertenencia se denota entonces como $\mu_A(x) \in [0, 1]$. [Maguiña 2010]

Por su parte, los sistemas de inferencia borrosos (SIB) son los encargados de manipular los conjuntos borrosos mediante las reglas que describen el conocimiento del proceso en cuestión y los términos lingüísticos definidos, utilizando modelos de inferencia como los de Mamdani, Sugeno y Tsukamoto [Ferreira 2014]. Estos sistemas son considerados sistemas expertos con un razonamiento aproximado para mapear un vector de entrada a una salida única basándose en la lógica difusa [Maguiña 2010].

El modelo propuesto por Sugeno se adapta mejor a los análisis matemáticos y resulta eficiente para términos computacionales, técnicas lineales, de optimización y adaptativas; este modelo es también conocido como Takagi, Takagi-Sugeno, Kang o TSK. En este modelo la conclusión de las reglas borrosas definidas es representada como una función lineal de las variables de entrada. Además, de no necesitar un proceso de concreción ya que cada regla tiene un valor exacto de salida a los cuales se les aplica un promedio o suma ponderada para obtener el resultado final. [Maguiña 2010]

El modelo propuesto por Tsukamoto es el menos utilizado debido a la poca transparencia que tiene respecto a los otros modelos. En este modelo la conclusión de las reglas borrosas definidas es representada a través de un conjunto borroso con una función de membresía monoatómica. Además, de definir un valor exacto para cada regla inducido por la fuerza de activación de la regla, por lo que la salida final del sistema de inferencia está dada por el promedio ponderado de cada salida de las reglas lo que implica no realizar un proceso de concreción. [Lee 1990]

El modelo propuesto por Mamdani ha sido más ampliamente aceptado al ser considerado más intuitivo y adaptable al lenguaje humano según [Maguiña 2010], además de poder ser convertido al tipo Sugeno según [Jassbi 2007]. Este método propone definir una base de reglas borrosas que contenga el siguiente formato: Si u_1 es A_1 y u_2 es A_2 y u_n es A_n entonces v es B, donde u_j y v son etiquetas lingüísticas, y A_j y B son los valores lingüísticos que dichas etiquetas pueden asumir.

Para obtener el conjunto borroso de salida en cada una de las reglas del SIB, se debe aplicar un operador T-normas que consiste en la intersección de los antecedentes o conjuntos borrosos de entrada; así como para obtener un único conjunto borroso de salida del SIB, se debe aplicar un operador T-conormas que consiste en la unión de cada uno de los consecuentes o conjuntos borrosos de salida. Las T-normas más utilizadas son: el mínimo, el producto algebraico, el producto acotado, el producto drástico, el producto de Hamacher y el producto de Einstein. Las T-conormas más utilizadas son: el máximo, la suma acotada y la suma algebraica. [Peregrín 2000]

La selección de los operadores a utilizar para realizar el razonamiento con un SIB dependerá de las necesidades de la situación que se analice. El operador más empleado entre las T-normas resulta el mínimo y entre las T-conormas el máximo. Esta situación se debe a que estos operadores son los que presentan menor sensibilidad extrema entre todos los operadores. La sensibilidad se define como el valor que mide el comportamiento de un operador al introducirse pequeños cambios en los valores de entrada. [La Red 2014]

Finalmente, para expresar mediante un valor nítido, los valores borrosos que se obtienen como salida del proceso de inferencia, se realiza un proceso de concreción. Los principales métodos que son utilizados para realizar el proceso de concreción de los datos son: Centro del área, Método de la altura, Media de máximo, Máximo más chico y Máximo más grande. La utilización del Método de la altura requiere la definición de un punto umbral y puede producir una salida no continua para pequeños cambios en la entrada. Por su parte los métodos que utilizan los máximos se recomienda utilizarlos cuando se tratan conjuntos borrosos con salidas simétricas. A pesar del costo computacional que se le atribuye al método Centro del área, por los cálculos que realiza, es considerado el método más potente y por tanto es el más utilizado. En la fórmula 3 se muestra la expresión que define al método Centro del área según [Bojórquez 2014].

$$COA = \frac{\sum_{z=a}^{b} \mu_A(z) \cdot z}{\sum_{z=a}^{b} \mu_A(z)} \qquad (3)$$

1.6. Computación con Palabras y el modelo de representación lingüística de 2-tuplas

La Computación con Palabras tiene sus inicios desde la definición de los conceptos de conjuntos borrosos y variables lingüísticas. Es una metodología de razonamiento, computación y toma de decisiones utilizando palabras del lenguaje natural en lugar de números. Con esta metodología se

logra realizar un razonamiento cuando la información disponible no es precisa aprovechando la tolerancia de la imprecisión para alcanzar una mejor relación con la realidad; además de proporcionar las bases necesarias para desarrollar lenguajes de programación que pueden acercarse a los lenguajes naturales. [Zadeh 2012]

La aplicación de la Computación con Palabras para la toma de decisiones lingüísticas ha crecido a través de los años al ser capaz de proporcionar herramientas cercanas a los procesos de razonamiento de los seres humanos. Pero su aplicación no se ha limitado solo a este campo, sino que también ha sido aplicada en el campo del aprendizaje, la clasificación y las bases de datos. Varios autores han propuesto diferentes modelos para llevar a cabo los procesos de esta metodología, entre estos se encuentra el modelo de representación lingüística de 2-tuplas. Este modelo tiene como objetivo mejorar la precisión de los resultados y facilitar los procesos de la Computación con Palabras al tratar el dominio lingüístico como un dominio continuo que mantiene su base lingüística, es decir, su sintaxis y semántica. [Zulueta 2014]

Este modelo de representación de información lingüística se basa en el concepto de traslación simbólica representando la información con dos valores a los que se denomina 2-tupla lingüística. El primer valor representa a uno de los términos lingüísticos definidos y el segundo es un valor numérico que representa la traslación simbólica del término lingüístico. La traslación simbólica de un término lingüístico s_i es definida por [Martínez 1999] como la diferencia de información que existe entre una cantidad de información expresada por el valor $\beta_i \in [0, g]$ obtenido en una operación simbólica y el valor entero más próximo, $i \in \{0,\ldots,g\}$, que indica el índice de la etiqueta lingüística (s_i) más cercana en el conjunto de términos lingüísticos S; esta diferencia es un número valorado en el intervalo [-0.5, 0.5).

A partir de este concepto [Martínez 1999] define una 2-tupla como el par (r_i, α_i), donde $r_i \in S = \{s_0,\ldots,s_g\}$ que representa a la etiqueta lingüística y $\alpha_i \in [-0.5, 0.5)$ que representa el número que expresa el valor de la distancia desde el resultado original $\beta_i \in [0, g]$ al índice de la etiqueta lingüística más cercana (r_i) en el conjunto de términos lingüísticos S. Para obtener la 2-tupla lingüística que expresa la información contenida en β se debe utilizar la fórmula 4.

$$\Delta : [0, g] \longrightarrow Sx[-.5, .5)$$

$$\Delta(\beta) = (s_i, \alpha), \ con \ \begin{cases} s_i, & i = round(\beta) \\ \alpha = \beta - i, & \alpha \in [-.5, .5), \end{cases} \tag{4}$$

Se debe señalar que Δ es biyectiva y que por lo tanto $\Delta^{-1}: \hat{S} \rightarrow [0, g]$ se define como $\Delta^{-1}(s_i, \alpha) = i + \alpha = \beta$; quedando la 2-tupla en $\hat{S}$ identificada con el valor números en el intervalo [0, g]. Teniendo en

cuenta esto, para convertir un término lingüístico en un valor 2-tupla lingüístico se debe añadir un valor 0 como traslación simbólica, quedando $s_i \in S \rightarrow (s_i, 0) \in \hat{S}$

De igual manera fue definido un modelo computacional lingüístico basado en funciones de transformación, donde se definieron los operadores de comparación, negación y agregación. Este modelo permite realizar un proceso sencillo y sin pérdida de información donde el resultado del proceso realizado siempre se encuentra expresado en el dominio lingüístico inicial, lo que permite mejorar la precisión de sus resultados al no tener que realizar operaciones de aproximación. A continuación se presentan los operadores definidos para este modelo, la elección del operador a utilizar en cada situación dependerá de las características específicas del problema que se pretende resolver. [Martínez 1999]

1.6.1. Comparación y negación de 2-tuplas

Para dos 2-tuplas lingüísticas, (s_i, α_1) y (s_k, α_2), al comparar la información lingüística que cada una representa, el resultado estaría dado por:

- Si $k < i$ entonces (s_k, α_2) es menor que (s_i, α_1)
- Si $k = i$ se deben comparar los valores α
    - Si $\alpha_1 < \alpha_2$ entonces (s_i, α_1) y (s_k, α_2) representan la misma información
    - Si $\alpha_1 < \alpha_2$ entonces (s_i, α_1) es menor que (s_k, α_2)
    - Si $\alpha_1 > \alpha_2$ entonces (s_i, α_1) es mayor que (s_k, α_2)

Para la negación de una 2-tupla fue definida la utilización de la siguiente expresión:

$$Neg((si, \alpha)) = \Delta(g - (\Delta^{-1}(si, \alpha)))$$

Donde $g + 1$ representa la cardinalidad de S y $S = \{s_0, \ldots, s_g\}$.

1.6.2. Agregación de 2-tuplas

El proceso de agregación consiste en obtener un valor colectivo que permita expresar la información de un conjunto de valores marginales. El resultado de aplicar uno de los operadores de agregación de información lingüística que han sido definidos se expresa en una 2-tupla. Para la definición de los operadores de agregación para el modelo de representación lingüística de 2-tuplas en [Martínez 1999], se tomó como base los operadores de agregación numéricos y simbólicos existentes. A continuación se presentan los operadores de agregación de información lingüística definidos.

1. LOWA extendido

Este operador se basa en la combinación convexa utilizando un vector de pesos. Donde para un conjunto de 2-tuplas A = {(r_1, α_1),…, (r_m, α_m)} y un vector de pesos asociado W = {w_1,…, w_m}, el operador LOWA extendido se define como:

$$\phi^e((r_1,\alpha_1),\ldots,(r_m,\alpha_m)) = W \cdot B^T = EC^m\{w_i,(r_{\sigma_{(i)}},\alpha_{\sigma_{(i)}}),i=1,\ldots,m\} \quad (5)$$

2. Media Aritmética Extendida

Este operador obtiene el punto de equilibrio o centro del conjunto de valores. Donde para un conjunto de 2-tuplas x = {(r_1, α_1),…, (r_m, α_m)}, la media aritmética extendida se define como:

$$\overline{x}^e((r_1,\alpha_1),\ldots,(r_m,\alpha_m)) = \Delta(\frac{1}{n}\sum_{i=1}^{n}\Delta^{-1}(r_i,\alpha_i)) = \Delta(\frac{1}{n}\sum_{i=1}^{n}\beta_i) \quad (6)$$

3. Media Ponderada Extendida

Con este operador se permite diferenciar los valores por su importancia, que se encuentra expresada en una 2-tupla lingüística. Donde para un conjunto de 2-tuplas x = {(r_1, α_1),…, (r_m, α_m)}, el vector de pesos lingüísticos asociado W = {(w_1, α_1),…, (w_m, α_m)} y $\beta_{wi} = \Delta^{-1}(w_i,\alpha_i)$, la media ponderada extendida se define como:

$$\overline{x}_l^e = \Delta(\frac{\sum_{i=1}^{m}\Delta^{-1}(r_i,\alpha_i)\cdot\Delta^{-1}(w_i,\alpha_i)}{\sum_{i=1}^{m}\Delta^{-1}(w_i,\alpha_i)}) = \Delta(\frac{\sum_{i=1}^{m}\beta_i\cdot\beta_{w_i}}{\sum_{i=1}^{m}\beta_{w_i}}) \quad (7)$$

4. OWA Extendido

Es un operador de agregación ponderado en el que los pesos no se encuentran asociados a un valor predeterminado, sino a una posición determinada. Donde para un conjunto de 2-tuplas A = {(r_1, α_1),…, (r_m, α_m)}, un vector de pesos asociado W = {w_1,…, w_m} que satisface que wi ∈ [0, 1] y $\sum w_i = 1$, siendo el β_j^* el j-ésimo mayor valor de los $\Delta^{-1}((r_i,\alpha_i))$, el operador OWA extendido se define como:

$$F^e((r_1,\alpha_1),\ldots,(r_m,\alpha_m)) = \Delta(\sum_{j=1}^{m} w_j\cdot\beta_j^*) \quad (8)$$

5. OR-LIKE S-OWA Extendido

Este operador es definido por una familia de pesos OWA. Donde para un conjunto de 2-tuplas A = {(r_1, α_1),…, (r_m, α_m)} y $\zeta \in [0,1]$, el operador OR-LIKE S-OWA extendido se define como:

$$\begin{aligned} F_{SO}^e((r_1,\alpha_1),\ldots,(r_m,\alpha_m)) &= \Delta(\zeta\cdot\max_i\{\Delta^{-1}(r_i,\alpha_i)\}+\frac{1}{m}(1-\zeta)\cdot\sum_i\Delta^{-1}(r_i,\alpha_i)) \\ &= \Delta(\zeta\cdot\max_i\{\beta_i\}+\frac{1}{m}(1-\zeta)\cdot\sum_i\beta_i). \end{aligned} \quad (9)$$

$$w_i = \begin{cases} \frac{1}{n}(1-\zeta)+\zeta, & i=1 \\ \frac{1}{n}(1-\zeta), & i=2,\ldots,n. \end{cases}$$

6. AND-LIKE S-OWA Extendido

Este operador es definido por una familia de pesos OWA. Donde para un conjunto de 2-tuplas A = $\{(r_1, \alpha_1),\ldots, (r_m, \alpha_m)\}$ y $\vartheta \in [0, 1]$, el operador AND-LIKE S-OWA extendido se define como:

$$F^e_{SA}((r_1,\alpha_1),...,(r_m,\alpha_m)) = \Delta(\vartheta \cdot \min_i\{\Delta^{-1}(r_i,\alpha_i)\} + \frac{1}{m}(1-\vartheta)\cdot\sum_i \Delta^{-1}(r_i,\alpha_i))$$

$$= \Delta(\vartheta \cdot \min_i\{\beta_i\} + \frac{1}{m}(1-\vartheta)\cdot\sum_i \beta_i). \quad (10)$$

$$w_i = \begin{cases} \frac{1}{n}(1-\vartheta), & i = n \\ \frac{1}{n}(1-\vartheta)+\vartheta, & i = 1,...,n-1. \end{cases}$$

1.7. Conclusiones parciales

Luego del estudio realizado sobre los principales conceptos asociados a la estimación del esfuerzo en un proyecto de desarrollo de software, los métodos de estimación de tamaño de software y la utilización de las técnicas de *soft computing* para mejorar la precisión de las estimaciones se concluye lo siguiente:

- ✓ Los métodos de estimación del esfuerzo, aplicando su definición original, en la industria de desarrollo de software utilizan rangos estrictos para la clasificación de sus componentes, lo que provoca la presencia de los llamados límites bruscos afectando la precisión de la estimación realizada.

- ✓ Las técnicas de *soft computing,* basadas fundamentalmente en la aplicación de la lógica difusa, han sido utilizadas con éxito en la estimación del esfuerzo de los proyectos de desarrollo de software pero, son difíciles de generalizar debido a que los niveles de clasificación no son suficiente para todos los casos y se requiere contar con datos previos para determinarlos.

- ✓ A pesar de que la Computación con Palabras usando el modelo de representación lingüística de 2-tuplas, ha obtenido resultados positivos en la solución de problemas en otros contextos de investigación que implican razonamientos similares al realizado para la estimación del esfuerzo en un proyecto de desarrollo de software, esta no ha sido utilizada para mejorar la precisión de las estimaciones en la industria del software.

CAPÍTULO 2: ALGORITMO PARA LA ESTIMACIÓN DEL ESFUERZO A PARTIR DE PUNTOS DE FUNCIÓN Y TÉCNICAS DE SOFT COMPUTING

Con el objetivo de obtener estimaciones del esfuerzo más precisas en un proyecto de desarrollo de software a partir de la determinación de la cantidad de puntos de función, en esta investigación se propone la utilización de dos algoritmos que permiten estimar el esfuerzo a partir de puntos de función y técnicas de *soft compunting*. En un primer momento, donde no se cuente con los registros históricos que permitan determinar la tendencia de desarrollo según la cantidad de puntos de función de un software, se estima el esfuerzo a partir de la aplicación de un algoritmo que utiliza el modelo de representación lingüístico de 2-tuplas.

A partir del momento en que se pueda contar con registros históricos sobre desarrollos previos en los que se pueda determinar la cantidad de puntos de función del software desarrollado, se propone la utilización de un segundo algoritmo que utiliza un Sistema de Inferencia Borroso. Para esto en las secciones del presente capítulo se describen los métodos de inferencia borrosa utilizados para determinar los pesos de los componentes; así como las operaciones de agregación realizadas en el modelo de representación lingüístico de 2-tuplas.

2.1. Algoritmo para estimar el esfuerzo a partir de puntos de función utilizando el modelo de representación lingüística de 2-tuplas

Para definir el algoritmo se tuvieron en cuenta las actividades que han sido realizadas por otros investigadores en la solución de problemas que implican razonamientos similares. A partir de este análisis se definieron once pasos a realizar, los cuales se muestran en la figura 2 y se describen en las secciones siguientes.

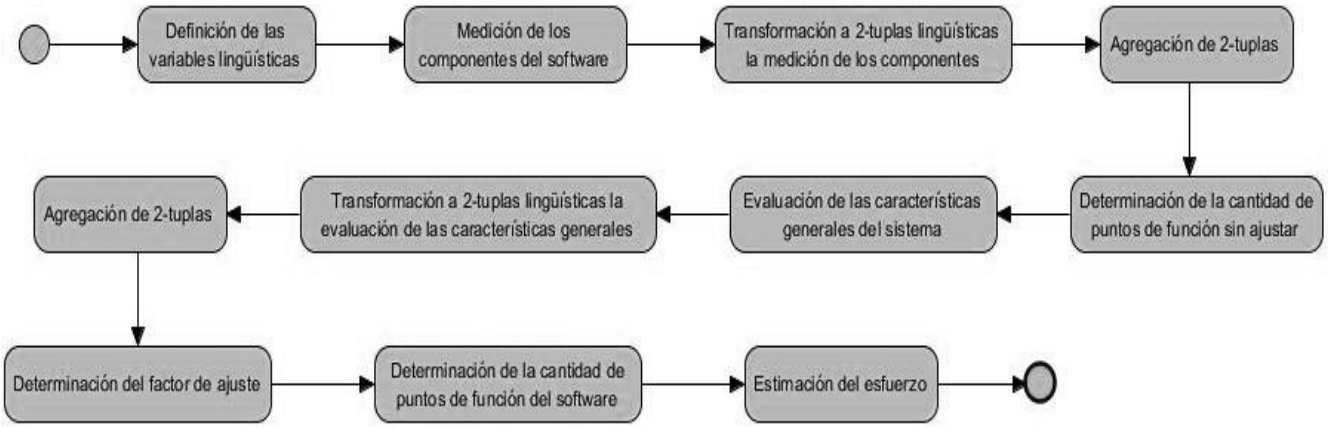


Figura 2. Algoritmo para estimar el esfuerzo a partir de puntos de función utilizando 2-tuplas

2.1.1. Paso 1: Definición de las variables lingüísticas a utilizar

Se definen como variables lingüísticas a utilizar en la determinación de los puntos de función sin ajustar los conjuntos: Tipo de Elementos de Datos (TED), Tipo de Elementos de Registros (TER),

Tipo de Archivo Referenciado (TAR), Puntos de Función (PF) y Grados de Influencia (GI). Las etiquetas utilizadas en las variables lingüísticas se encuentran simétricamente divididas.

Para las etiquetas lingüísticas definidas para las variables lingüísticas TED, TER, TAR y PF están dadas por: MB = Muy Bajo, B = Bajo, M = Medio, A = Alto y MA = Muy Alto. Así como para la variable lingüística GI se definieron las etiquetas lingüísticas: SI = Sin Influencia, AC = Accidental, MO = Moderado, ME = Medio, SG = Significativo y ES = Esencial.

De esta forma el conjunto de términos lingüísticos utilizado en las variables lingüísticas TED, TER, TAR y PF, se encuentra formado por una estructura de etiquetas ordenadas de la siguiente manera:

$$S = \{ s_0 = MB, s_1 = B, s_2 = M, s_3 = A, s_4 = MA \} \text{ con } s_i < s_j \text{ para } i < j$$

De esta forma el conjunto de términos lingüísticos utilizado en la variable lingüística GI se encuentra formado por una estructura de etiquetas ordenadas de la siguiente manera:

$$S = \{ s_0 = SI, s_1 = AC, s_2 = MO, s_3 = ME, s_4 = SG, s_5 = ES \} \text{ con } s_i < s_j \text{ para } i < j$$

Las variables lingüísticas TER, TED y PF, son utilizadas para proporcionar los puntos de función sin ajustar de los componentes del método Archivos internos lógicos (AIL) y Archivos de interfaz externos (AIE). Mientras que para los componentes Entrada externa (EE), Salida externa (SE) y Consulta externa (CE) serán utilizadas las variables lingüísticas TED, TAD y PF. Además, para determinar el factor de ajuste a aplicar se utilizará la variable lingüística GI.

El conjunto TED representa al número de campos únicos que son reconocidos por el usuario. El conjunto TER representa al número de subgrupos de elementos de datos que son reconocidos por el usuario. El conjunto TAR representa al número de AIL o AIE que se han sido leídos y mantenidos en una función transaccional. El conjunto PF representa el peso que se le otorgará para la estimación de acuerdo a la cantidad de TER y TED o TAR y TED del componente que se analice. Así como, el conjunto GI representa los grados de influencia de cada una de las catorce características generales a evaluar en el sistema a desarrollar que propone el método Puntos de Función. [IFPUG 2010]

La representación de las funciones de pertenencia de cada una de las etiquetas lingüísticas definidas en esta investigación se realizó a través del modelo matemático función triangular. A continuación se muestran las funciones de pertenencia asociadas a cada una de las etiquetas de las variables lingüísticas definidas, TED, TER, TAR y GI.

<u>TED, TER, TAR</u>
Muy Bajo: Triángulo (x, nulo, 0, 1)
Bajo: Triángulo (x, 0, 1, 2)
Medio: Triángulo (x, 1, 2, 3)
Alto: Triángulo (x, 2, 3, 4)
Muy Alto: Triángulo (x, 3, 4, nulo)

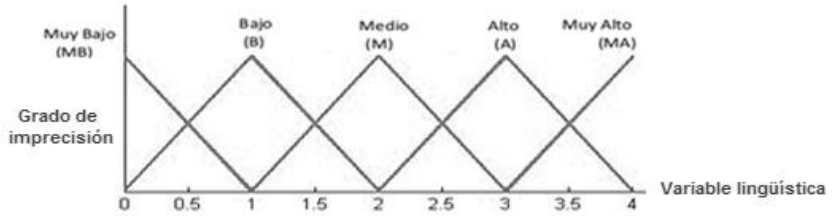


Figura 3. Etiquetas lingüísticas para la evaluación de los TED, TER y TAR con 2-tuplas

Grados de Influencia

Sin Influencia: Triángulo (x, nulo, 0, 1)
Accidental: Triángulo (x, 0, 1, 2)
Moderado: Triángulo (x, 1, 2, 3)
Medio: Triángulo (x, 2, 3, 4)
Significativo: Triángulo (x, 3, 4, 5)
Esencial: Triángulo (x, 4, 5, nulo)

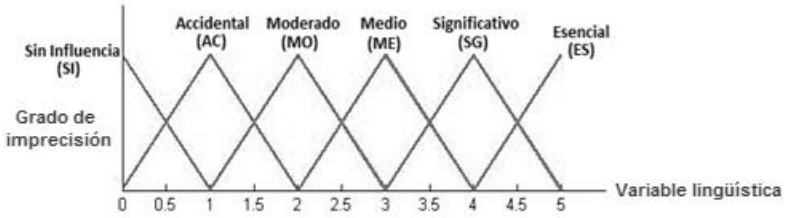


Figura 4. Etiquetas lingüísticas para la evaluación de los GI con 2-tuplas

Teniendo en cuenta que en el método tradicional el dominio de los puntos de función que se otorgan para la estimación de acuerdo a la cantidad de TER y TED o TAR y TED no son iguales en todos los componentes, se convierte a esta variable lingüística en cuatro variables lingüísticas: PF_{AIL}, PF_{AIE}, $PF_{EE\text{-}CE}$ y PF_{SE}. Las etiquetas lingüísticas de estas variables son las mismas que se utilizan en las variables lingüísticas TED, TER, TAR y GI. A continuación se muestran las funciones de pertenencia asociadas a cada una de las etiquetas de las variables lingüísticas PF_{AIL}, PF_{AIE}, $PF_{EE\text{-}CE}$ y PF_{SE}.

Puntos de Función para AIL

PF Muy Bajo: Triángulo (x, 6, 7, 8)
PF Bajo: Triángulo (x, 8, 9, 10)
PF Medio: Triángulo (x, 10, 11, 12)
PF Alto: Triángulo (x, 12, 13, 14)
PF Muy Alto: Triángulo (x, 14, 15, 16)

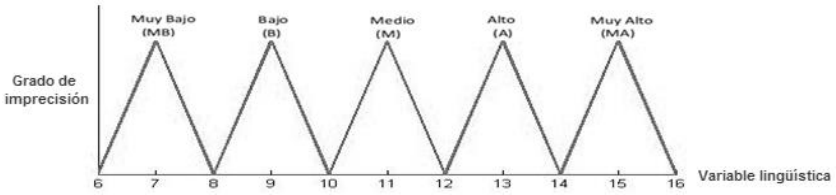


Figura 5. Etiquetas lingüísticas para la evaluación de los PF con 2-tuplas para los AIL

Puntos de Función para AIE

PF Muy Bajo: Triángulo (x, 4.375, 5, 5.625)

PF Bajo: Triángulo (x, 5.625, 6.25, 6.875)

PF Medio: Triángulo (x, 6.875, 7.5, 8.125)

PF Alto: Triángulo (x, 8.125, 8.75, 9.375)

PF Muy Alto: Triángulo (x, 9.375, 10, 10.625)

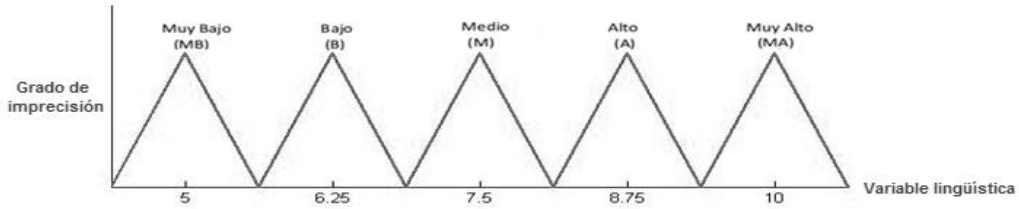


Figura 6. Etiquetas lingüísticas para la evaluación de los PF con 2-tuplas para los AIE

Puntos de Función para EE y CE

PF Muy Bajo: Triángulo (x, 2.625, 3, 3.375)

PF Bajo: Triángulo (x, 3.375, 3.75, 4.125)

PF Medio: Triángulo (x, 4.125, 4.5, 4.875)

PF Alto: Triángulo (x, 4.875, 5.25, 5.625)

PF Muy Alto: Triángulo (x, 5.625, 6, 6.375)

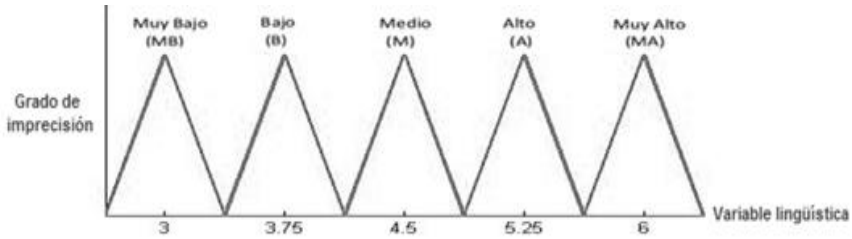


Figura 7. Etiquetas lingüísticas para la evaluación de los PF con 2-tuplas para las EE y CE

Puntos de Función para SE

PF Muy Bajo: Triángulo (x, 6, 7, 8)

PF Bajo: Triángulo (x, 8, 9, 10)

PF Medio: Triángulo (x, 10, 11, 12)

PF Alto: Triángulo (x, 12, 13, 14)

PF Muy Alto: Triángulo (x, 14, 15, 16)

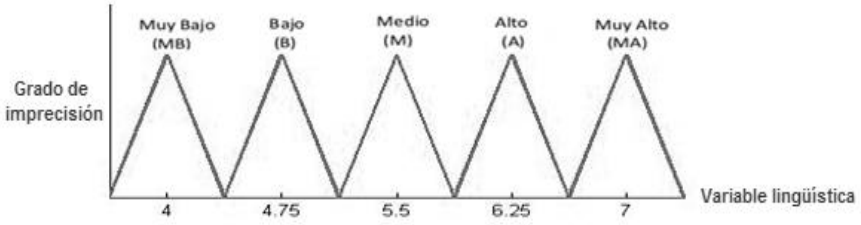


Figura 8. Etiquetas lingüísticas para la evaluación de los PF con 2-tuplas para las SE

2.1.2. Paso 2: Medición de los componentes del software

Para realizar la medición del software que se estima teniendo en cuenta la documentación generada hasta el momento sobre las funcionalidades que deben ser desarrolladas, se identifican los componentes AIL, AIE, EE, SE y CE del software a desarrollar. Luego se realiza el conteo de las funciones de datos y transaccionales de cada uno de los componentes que fueron identificados para obtener la cantidad de TED, TER y TAR de cada uno. Para realizar un conteo correcto de las funciones de datos y transaccionales en cada uno de los componentes deben ser aplicadas todas las aclaraciones, reglas, guías y criterios que se encuentran definidas en el manual de prácticas de conteo en [IFPUG 2010]. En la tabla 15 se muestra un ejemplo de los datos que se esperan obtener al finalizar este paso.

Tabla 15: Conteo de las funciones de datos y transaccionales de un software

Componentes	Cantidad	TED	TER / TAR
Entradas Externas	2	4	2
	2	9	3
	2	19	4
Consultas Externas	1	4	2
	1	11	4
	1	21	5
Salidas Externas	3	5	1
	2	12	3
	2	19	4
Archivos Internos Lógicos	4	10	2
	2	20	4
	1	38	5
Archivos de Interfaz Externos	1	10	2
	1	20	2
	1	52	5

2.1.3. Paso 3: Transformación a 2-tuplas lingüísticas la medición de los componentes

Cada valor de TED, TER y TAR obtenidos en el paso anterior en cada uno de los componentes debe ser transformado en una 2-tupla lingüística. Según [Martínez 1999] una 2-tupla se encuentra definida como el par (r_i, α_i), donde $r_i \in S = \{s_0,\ldots,s_g\}$ que representa a la etiqueta lingüística y $\alpha_i \in [-0.5, 0.5)$ que representa el número que expresa el valor de la distancia desde el resultado original $\beta_i \in [0, g]$ al índice de la etiqueta lingüística más cercana (r_i) en el conjunto de términos lingüísticos S; donde el valor $\beta_i \in [0, g]$ es obtenido en una operación simbólica.

Teniendo en cuenta lo anterior en esta investigación se propone la utilización de la fórmula 11 para determinar el valor β_i de forma tal que se tenga en cuenta las diferencias de los dominios de las

variables lingüísticas utilizadas. Donde la variable C_{vl} representa el valor de la medición realizada al sistema que se estime para la variable lingüística que se calcule. La variable CM_{vl} representa el valor máximo según el método de Puntos de Función que puede ser medido para la variable lingüística que se calcule. La variable IM_{vl} representa al mayor índice definido en el conjunto de términos lingüísticos de la variable lingüística que se calcule.

$$\beta_i = \frac{C_{vl}}{CM_{vl}} * IM_{vl} \qquad (11)$$

Luego de obtener el valor β_i para cada TED, TER y TAR, se utiliza la fórmula 3 definida por [Martínez 1999] para transformar cada uno de estos valores en la correspondiente 2-tupla lingüística. En la tabla 16 se muestra la transformación de cada uno de los datos obtenidos en el paso 2 del algoritmo.

Tabla 16: Transformación a 2-tuplas lingüísticas la medición de los componentes Conteo de las funciones de datos y transaccionales de un software

Componentes	TED	TER / TAR	2-tuplas TED	2-tuplas TER / TAR
Entradas Externas	4	2	(S_1, 0)	(S_3, -0.33)
	9	3	(S_2, 0.25)	(S_4, 0)
	16	3	(S_4, 0)	(S_4, 0)
Consultas Externas	4	2	(S_1, -0.20)	(S_2, 0)
	11	4	(S_2, 0.20)	(S_4, 0)
	20	4	(S_4, 0)	(S_4, 0)
Salidas Externas	5	1	(S_1, 0)	(S_1, 0)
	12	3	(S_2, 0.40)	(S_3, 0)
	19	4	(S_4, -0.20)	(S_4, 0)
Archivos Internos Lógicos	10	2	(S_1, -0.22)	(S_1, 0.33)
	20	4	(S_2, -0.43)	(S_3, -0.33)
	38	5	(S_3, -0.02)	(S_3, 0.33)
Archivos de Interfaz Externos	10	2	(S_1, -0.22)	(S_1, 0.33)
	20	2	(S_2, -0.43)	(S_1, 0.33)
	49	5	(S_4, -0.16)	(S_3, 0.33)

2.1.4. Paso 4: Agregación de 2-tuplas

Para transformar las 2-tuplas de las variables lingüísticas TER y TED o TAR y TED, según el componente que sea analizado, en un conjunto de valores de preferencia colectiva de cada uno de los componentes, es necesario realizar un proceso de agregación. Luego del estudio realizado en esta investigación sobre los operadores de agregación que se han definido para el modelo de representación lingüística de 2-tuplas, fue seleccionado el operador Media Aritmética Extendida que se muestra en la fórmula 5 según [Martínez 1999] lo define. Los resultados alcanzados al realizar la operación de agregación entre las 2-tuplas obtenidas en el ejemplo, se muestran en la tabla 17.

Tabla 17: Agregación de 2-tuplas para el ejemplo

Componentes	2-tuplas TED	2-tuplas TER / TAR	$\bar{x}^e$
Entradas Externas	(S_1, 0)	(S_3, -0.33)	(S_2, -0.17)
	(S_2, 0.25)	(S_4, 0)	(S_3, 0.13)
	(S_4, 0)	(S_4, 0)	(S_4, 0)
Consultas Externas	(S_1, -0.20)	(S_2, 0)	(S_1, 0.40)
	(S_2, 0.20)	(S_4, 0)	(S_3, 0.10)
	(S_4, 0)	(S_4, 0)	(S_4, 0)
Salidas Externas	(S_1, 0)	(S_4, 0)	(S_4, 0)
	(S_2, 0.40)	(S_3, 0)	(S_3, -0.30)
	(S_4, -0.20)	(S_4, 0)	(S_4, -0.10)
Archivos Internos Lógicos	(S_1, -0.22)	(S_1, 0.33)	(S_1, 0.06)
	(S_2, -0.43)	(S_3, -0.33)	(S_2, 0.12)
	(S_3, -0.02)	(S_3, 0.33)	(S_3, 0.16)
Archivos de Interfaz Externos	(S_1, -0.22)	(S_1, 0.33)	(S_1, 0.06)
	(S_2, -0.43)	(S_1, 0.33)	(S_1, 0.45)
	(S_4, -0.16)	(S_3, 0.33)	(S_4, -0.41)

2.1.5. Paso 5: Determinación de la cantidad de puntos de función sin ajustar

La cantidad de puntos de función sin ajustar del sistema a desarrollar está dada por la sumatoria de los puntos de función sin ajustar asignados a cada uno de los componentes que comprenda. Para determinar un valor nítido de puntos de función sin ajustar que represente la conclusión cualitativa de la variable lingüística que se analice en cada uno de los componentes medidos, se propone sumarle al peso que le sea asignado al componente ($\mathrm{PF_n}$), la multiplicación del valor de α_{in} por la diferencia que exista entre los valores nítidos que tiene el grado de pertinencia igual a uno en el conjunto borroso que representa los puntos de función del componente que se analiza (d).

El peso asignado a cada componente medido ($\mathrm{PF_n}$) se corresponde al valor nítido que tiene el grado de pertinencia igual a uno en el conjunto borroso correspondiente al índice del término lingüístico obtenido con el operador de agregación en el paso 4 del algoritmo. Este conjunto borroso representa a través de una función de pertenencia triangular la semántica de la variable lingüística PF. Teniendo en cuenta lo anterior, en esta investigación se utiliza la fórmula 12 para determinar los puntos de función sin ajustar. En la tabla 18 se muestra la cantidad de puntos de función sin ajustar para el ejemplo que se describe en los pasos del algoritmo. En la columna PFSA 2-tuplas Total, se muestra el resultado de multiplicar la cantidad de componentes que tienen la misma cantidad de TED y TER/TAR, con la cantidad de puntos de función que fueron calculados por el algoritmo que se le deben asignar a un componente con estas características. Además se muestran los puntos de función sin ajustar que le serían asignados a cada uno de los componentes al utilizar el método Puntos de Función en su forma tradicional.

Tabla 18: Puntos de función sin ajustar calculados para cada componente del ejemplo

Componentes	**Cantidad**	$\bar{x}^e$	PF_n	d	α_{in}	***PFSA 2-tuplas***	***PFSA 2-tuplas total***	***PFSA tradicional***
Entradas Externas	2	(S_2, -0.17)	4.5	0.75	-0.17	4.37	8.74	6
	2	(S_3, 0.13)	5.25		0.13	5.35	10.70	12
	2	(S_4, 0)	6		0	6	12	12
Consultas Externas	1	(S_1, 0.40)	3.75	0.75	0.40	4.05	4.05	3
	1	(S_3, 0.10)	5.25		0.10	5.33	5.33	6
	1	(S_4, 0)	6		0	6	6	6
Salidas Externas	3	(S_4, 0)	7	0.75	0	7	21	12
	2	(S_3, -0.30)	6.25		-0.30	6.01	12.02	10
	2	(S_4, -0.10)	7		-0.10	6.93	13.86	14
Archivos Internos Lógicos	4	(S_1, 0.06)	9	2	0.06	9.12	36.48	28
	2	(S_2, 0.12)	11		0.12	11.24	22.48	20
	1	(S_3, 0.16)	13		0.16	13.32	13.32	10
Archivos de Interfaz Externos	1	(S_1, 0.06)	6.25	1.25	0.06	6.33	6.33	5
	1	(S_1, 0.45)	6.25		0.45	6.81	6.81	7
	1	(S_4, -0.41)	10		-0.41	9.49	9.49	7

$$\text{PFSA} = \sum_1^n \text{PF}_\text{n} + d * \alpha_\text{in} \qquad (12)$$

En caso de estimar un software con las características del ejemplo que ha sido utilizado en los pasos del algoritmo propuesto, entonces, sus puntos de función sin ajustar están dados por:

$$\text{PFSA} = \text{PFSA}_1 + \text{PFSA}_2 + \ldots + \text{PFSA}_{15} = 8.74 + 10.70 + \ldots + 9.49 = 188.61 \qquad (13)$$

Al utilizarse el método tradicional de Puntos de Función, para el mismo escenario calculado anteriormente, se obtienen 158 puntos de función sin ajustar. Con estos valores se obtiene entonces una diferencia de 30.61 puntos de función entre los métodos de estimación. Esta cantidad representa una diferencia de 352 horas si se toma la tendencia que existe de asignar 11.50 horas por cada punto de función, lo cual representa una diferencia de nueve semanas para el proceso de desarrollo de software.

2.1.6. Paso 6: Evaluación de las características generales del sistema

Para realizar la evaluación de las 14 características generales del sistema a desarrollar, se recopilaron los criterios de los expertos disponibles en el proyecto de desarrollo de software. Aunque es suficiente con el criterio que sea emitido por una sola persona, se recomienda utilizar la opinión de varios expertos. En la tabla 19 se muestran las criterios que fueron emitidos por dos expertos para cada una de las características generales del sistema que se ejemplifica en el algoritmo.

Tabla 19: Evaluación de las características generales de un sistema

Número	Característica	Criterio Experto 1	Criterio Experto 2
1.	Comunicación de datos	Moderado	Medio
2.	Procesamiento distribuido de datos	Accidental	Accidental
3.	Rendimiento	Significativo	Significativo
4.	Configuración altamente utilizada	Medio	Moderado
5.	Promedio de transacciones	Medio	Esencial
6.	Entrada de datos en línea	Accidental	Accidental
7.	Eficiencia para el usuario final	Medio	Moderado
8.	Actualización en línea	Accidental	Accidental
9.	Procesamiento complejo	Moderado	Significativo
10.	Reusabilidad	Medio	Moderado
11.	Facilidad de instalación	Moderado	Significativo
12.	Facilidad de operación	Medio	Medio
13.	Múltiples localizaciones	Sin Influencia	Sin Influencia
14.	Facilidad de cambios	Esencial	Medio

2.1.7. Paso 7: Transformación a 2-tuplas lingüísticas la evaluación de las características generales

Dado que todos los criterios fueron emitidos sobre el mismo conjunto de etiquetas lingüísticas, la transformación de los criterios es realizada de manera directa, donde el valor de la traslación simbólica es cero. Por lo que la 2-tupla lingüística de la característica general p, quedaría como (Sp, 0). En la tabla 20 se muestra el resultado obtenido para cada criterio emitido por los expertos al transformarlo en una 2-tupla lingüística.

Tabla 20: Evaluación de las características generales de un sistema en 2-tuplas lingüísticas

Número	Característica	Criterio Experto 1	Criterio Experto 2
1.	Comunicación de datos	$(S_2, 0)$	$(S_3, 0)$
2.	Procesamiento distribuido de datos	$(S_1, 0)$	$(S_1, 0)$
3.	Rendimiento	$(S_4, 0)$	$(S_4, 0)$
4.	Configuración altamente utilizada	$(S_3, 0)$	$(S_2, 0)$
5.	Promedio de transacciones	$(S_3, 0)$	$(S_5, 0)$
6.	Entrada de datos en línea	$(S_1, 0)$	$(S_1, 0)$
7.	Eficiencia para el usuario final	$(S_3, 0)$	$(S_2, 0)$
8.	Actualización en línea	$(S_1, 0)$	$(S_1, 0)$
9.	Procesamiento complejo	$(S_2, 0)$	$(S_4, 0)$
10.	Reusabilidad	$(S_3, 0)$	$(S_2, 0)$
11.	Facilidad de instalación	$(S_2, 0)$	$(S_4, 0)$
12.	Facilidad de operación	$(S_3, 0)$	$(S_3, 0)$
13.	Múltiples localizaciones	$(S_0, 0)$	$(S_0, 0)$
14.	Facilidad de cambios	$(S_5, 0)$	$(S_3, 0)$

2.1.8. Paso 8: Agregación de 2-tuplas

Con el objetivo de obtener los criterios colectivos a partir de los criterios individuales emitidos en cada una de las características generales del sistema para determinar el factor de ajuste a aplicar, se decidió utilizar el operador de agregación para 2-tuplas lingüísticas Media Aritmética Extendida definido por la fórmula 5. Con la fórmula 14 se realiza el cálculo para la primera característica general del sistema y luego se muestran en la tabla 21, las 2-tuplas lingüísticas obtenidas como resultado de aplicar el operador de agregación en cada una de las características generales del sistema.

$$\bar{x}_1^e = \Delta\left(\frac{1}{2}(2+3)\right) = \Delta(2.5) = (S_2, 0.5) \qquad (14)$$

Tabla 21: Criterios colectivos de las características generales de un sistema en 2-tuplas lingüísticas

Número	Característica	Criterio Colectivo $\bar{\mathbf{x}}_{\mathbf{n}}^{\mathbf{e}}$
1.	Comunicación de datos	(S_2, 0.5)
2.	Procesamiento distribuido de datos	(S_1, 0)
3.	Rendimiento	(S_4, 0)
4.	Configuración altamente utilizada	(S_2, 0.5)
5.	Promedio de transacciones	(S_4, 0)
6.	Entrada de datos en línea	(S_1, 0)
7.	Eficiencia para el usuario final	(S_2, 0.5)
8.	Actualización en línea	(S_1, 0)
9.	Procesamiento complejo	(S_3, 0)
10.	Reusabilidad	(S_2, 0.5)
11.	Facilidad de instalación	(S_3, 0)
12.	Facilidad de operación	(S_3, 0)
13.	Múltiples localizaciones	(S_0, 0)
14.	Facilidad de cambios	(S_4, 0)

Luego de obtener los criterios colectivos de cada una de las características generales del sistema se procede a aplicar nuevamente el operador de agregación Media Aritmética Extendida entre las catorce 2-tuplas que se obtienen de la variable lingüística GI. Este resultado representa la complejidad asociada al desarrollo del software según sus especificidades. A través de la fórmula 15 se muestra la operación de agregación realizada.

$$\bar{x}^e = \Delta\left(\frac{1}{14}(2+1+4+2+4+1+2+1+3+2+3+3+0+4)\right) = \Delta(2.29) = (S_2, 0.29) \qquad (15)$$

2.1.9. Paso 9: Determinación del factor de ajuste

Para determinar el factor de ajuste se utiliza la fórmula 16, en la cual se modifica uno de los coeficientes utilizados para el ajuste de la complejidad técnica del sistema en la fórmula del método tradicional con el objetivo de aplicar el modelo de representación lingüística de 2-tuplas. El valor β_{in} representa la complejidad de las catorce características generales del sistema a desarrollar que fueron evaluadas, transformadas y agregadas en los pasos 6, 7 y 8 de la propuesta de este algoritmo.

$$FA = 0.65 + 0.14 * \beta_{in} \quad (16)$$

La modificación propuesta se debe a que el valor utilizado en la fórmula tradicional se corresponde a la sumatoria de las evaluaciones obtenidas en cada una de las características y en el algoritmo propuesto el valor β_{in} que se obtiene expresa el centro del conjunto de valores de las evaluaciones obtenidas en cada una de las características. A través de la fórmula 17 se muestra el cálculo que se realiza para determinar el factor de ajuste del ejemplo utilizado.

$$FA = 0.65 + 0.14 * 2.29 = 0.97 \quad (17)$$

2.1.10. Paso 10: Determinación de la cantidad de puntos de función del software

Luego de calculados los puntos de función sin ajustar en el paso cinco y el factor de ajuste en el paso nueve, solo queda realizar la multiplicación de estos dos valores para obtener los puntos de función a utilizar en la estimación del tamaño del software a desarrollar. En la fórmula 18 se muestra el cálculo que se realiza para determinar la cantidad de puntos de función del software que se estima.

$$PF = 188.61 * 0.97 = 182.95 \quad (18)$$

2.1.11. Paso 11: Estimación del esfuerzo

Para estimar el esfuerzo que se le debe dedicar al desarrollo del software analizado como ejemplo en los pasos del algoritmo propuesto, se utilizó el valor medio de la industria, definido por el ISBSG, de 11.50 horas-hombre por punto de función. En la fórmula 19 se muestra el cálculo que se realiza para determinar la cantidad horas que se le deben dedicar al desarrollo del software que se estima.

$$Esfuerzo = 182.95 * 11.50 = 2\,104 \text{ horas} \quad (19)$$

2.2. Algoritmo para estimar el esfuerzo a partir de puntos de función utilizando un Sistema de Inferencia Borroso

Para definir el algoritmo se tuvieron en cuenta las actividades que han sido realizadas por otros investigadores en la solución de problemas utilizando un sistema de inferencia borroso. A partir de este análisis se definieron nueve pasos a realizar, los cuales se muestran en la figura 9 y se describen en las secciones que le siguen.

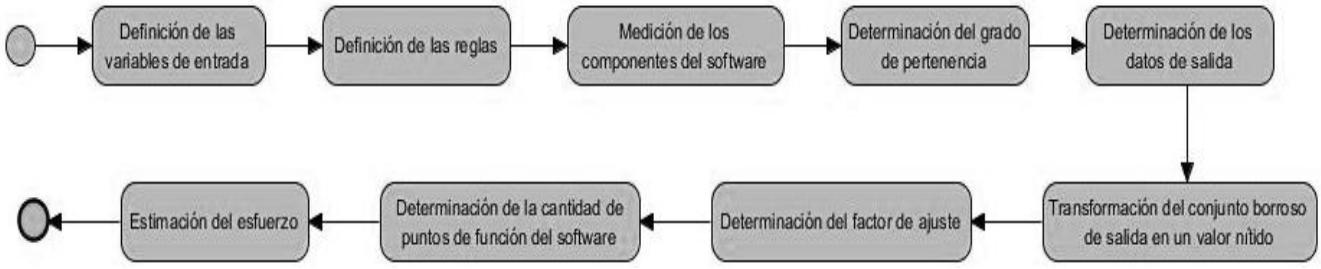


Figura 9. Algoritmo para estimar el esfuerzo a partir de puntos de función utilizando 2-tuplas

2.2.1. Paso 1: Definición de las variables de entrada

Las variables de entrada utilizadas corresponden a cada uno de los componentes que se definen en el método Puntos de Función, las cuales son:

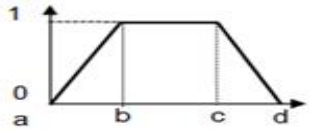
- Archivos internos lógicos (AIL)
- Archivos de interfaz externos (AIE)
- Entrada externa (EE)
- Salida externa (SE)
- Consulta externa (CE)

Para cada una de estas variables definidas se especifica su dominio, las etiquetas lingüísticas que los representan y sus conjuntos borrosos. Para las variables AIL y AIE se definen tres etiquetas lingüísticas: Tipo de Elementos de Datos (TED), Tipo de Elementos de Registros (TER) y Puntos de Función (PF). Para las variables EE, SE y CE se definen tres etiquetas lingüísticas: Tipo de Elementos de Datos (TED), Tipo de Archivo Referenciado (TAR) y Puntos de Función (PF).

El conjunto TED representa al número de campos únicos que son reconocidos por el usuario. El conjunto TER representa al número de subgrupos de elementos de datos que son reconocidos por el usuario. El conjunto TAR representa al número de AIL o AIE que son leídos y mantenidos en una función transaccional. El conjunto PF representa el peso que se otorga para la estimación de acuerdo a la cantidad de TER y TED o TAR y TED de la variable de entrada analizada. [IFPUG 2010]

Para la etiqueta lingüística TED, debido al amplio rango de valores que pertenecen a dicho conjunto, se propuso su evaluación con los términos Muy Bajo, Bajo, Medio, Alto y Muy Alto. A las restantes etiquetas lingüísticas se propuso evaluarlas con los términos Bajo, Medio y Alto según los mismos valores que propone el método Puntos de Función y que varían en cada una de las variables de entrada definidas. La representación de las funciones de pertenencia de cada una de las etiquetas lingüísticas definidas en esta investigación se realiza a través de los modelos matemáticos funciones trapezoidales y triangulares, según se muestra en la figura 10. A continuación se definen cada uno de los conjuntos borrosos a utilizar en cada una de las variables de entrada declaradas y se muestra su representación en las figuras del número 11 al número 25.

$$Trapecio\ (x,a,b,c,d) = \begin{cases} 0 & x \le a \\ \frac{x-a}{b-a} & a \le x \le b \\ 1 & b \le x \le c \\ \frac{d-x}{d-c} & c \le x \le d \\ 0 & x \ge d \end{cases}$$



$$Triángulo\ (x,a,b,c) = \begin{cases} 0 & x \le a \\ \frac{x-a}{b-a} & a \le x \le b \\ \frac{c-x}{c-b} & b \le x \le c \\ 0 & c \le x \end{cases}$$

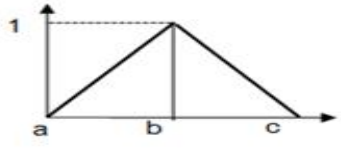


Figura 10. Funciones trapezoidal y triangular [Piñero 2005]

Archivos internos lógicos

TED Muy Bajo: Trapecio (x, nulo, 1, 6, 10)

TED Bajo: Trapecio (x, 3, 8, 12, 20)

TED Medio: Trapecio (x, 10, 20, 36, 43)

TED Alto: Trapecio (x, 28, 38, 44, 54)

TED Muy Alto: Trapecio (x, 43, 50, nulo, nulo)

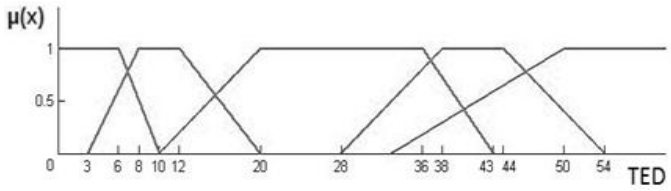


Figura 11. Conjuntos borrosos propuestos para la evaluación de la cantidad de TED en un AIL

TER Bajo: Trapecio (x, nulo, 1, 2, 4)

TER Medio: Trapecio (x, 1, 3, 5, 6)

TER Alto: Trapecio (x, 3, 6, nulo, nulo)

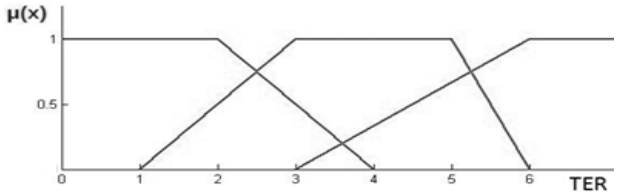


Figura 12. Conjuntos borrosos propuestos para la evaluación de la cantidad de TER en un AIL

PF Muy Bajo: Triángulo (x, 0, 1.8, 3.6)

PF Bajo: Triángulo (x, 3.6, 5.4, 7.2)

PF Medio: Triángulo (x, 7.2, 9, 10.8)

PF Alto: Triángulo (x, 10.8, 12.6, 14.4)

PF Muy Alto: Triángulo (x, 14.4, 16.2, 18)

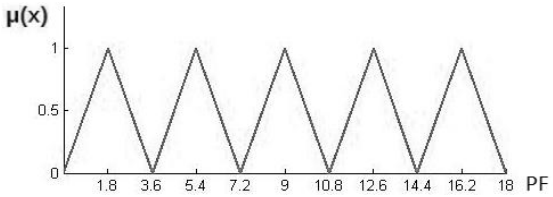


Figura 13. Conjuntos borrosos propuestos para la evaluación del peso de los PF en un AIL

Archivos de interfaz externos

TED Muy Bajo: Trapecio (x, nulo, 1, 6, 10)

TED Bajo: Trapecio (x, 3, 8, 12, 20)

TED Medio: Trapecio (x, 10, 20, 36, 43)

TED Alto: Trapecio (x, 28, 38, 44, 54)

TED Muy Alto: Trapecio (x, 43, 50, nulo, nulo)

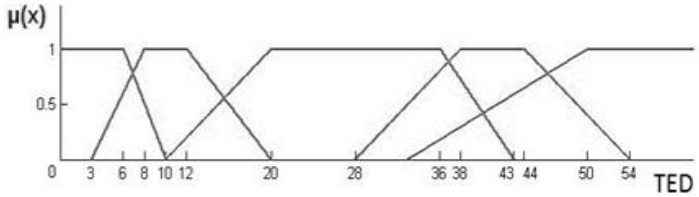


Figura 14. Conjuntos borrosos propuestos para la evaluación de la cantidad de TED en un AIE

TER Bajo: Trapecio (x, nulo, 1, 2, 4)
TER Medio: Trapecio (x, 1, 3, 5, 6)
TER Alto: Trapecio (x, 3, 6, nulo, nulo)

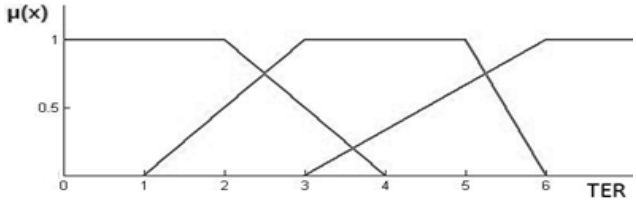


Figura 15. Conjuntos borrosos propuestos para la evaluación de la cantidad de TER en un AIE

PF Muy Bajo: Triángulo (x, 0, 1.3, 2.6)
PF Bajo: Triángulo (x, 2.6, 3.9, 5.2)
PF Medio: Triángulo (x, 5.2, 6.5, 7.8)
PF Alto: Triángulo (x, 7.8, 9.1, 10.4)
PF Muy Alto: Triángulo (x, 10.4, 11.7, 13)

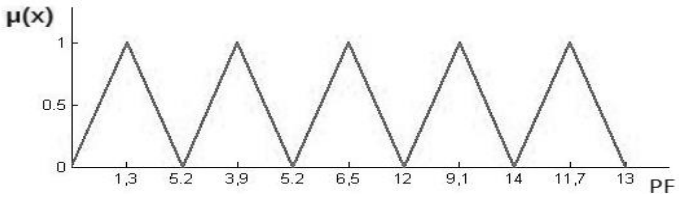


Figura 16. Conjuntos borrosos propuestos para la evaluación del peso de los PF en un AIE

<u>Entrada externa (EE)</u>

TED Muy Bajo: Trapecio (x, nulo, 1, 3, 7)

TED Bajo: Trapecio (x, 1, 5, 7, 11)

TED Medio: Trapecio (x, 6, 9, 11, 16)

TED Alto: Trapecio (x, 10, 14, 18, 22)

TED Muy Alto: Trapecio (x, 16, 22, nulo, nulo)

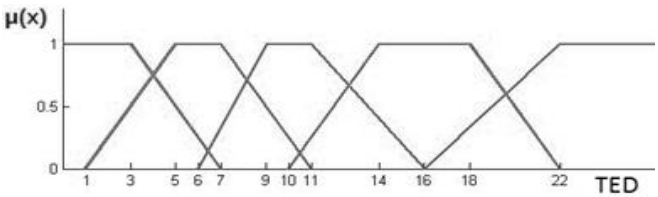


Figura 17. Conjuntos borrosos propuestos para la evaluación de la cantidad de TED en una EE

TAR Bajo: Trapecio (x, nulo, 0, 1, 2)
TAR Medio: Trapecio (x, 0, 2, 3, 4)
TAR Alto: Trapecio (x, 1, 3, nulo, nulo)

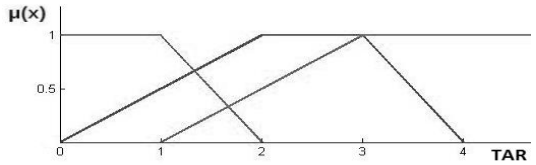


Figura 18. Conjuntos borrosos propuestos para la evaluación de la cantidad de TAR en una EE

PF Muy Bajo: Triángulo (x, 0, 0.8, 1.6)
PF Bajo: Triángulo (x, 1.6, 2.4, 3.2)
PF Medio: Triángulo (x, 3.2, 4, 4.8)
PF Alto: Triángulo (x, 4.8, 5.6, 6.4)
PF Muy Alto: Triángulo (x, 6.4, 7.2, 8)

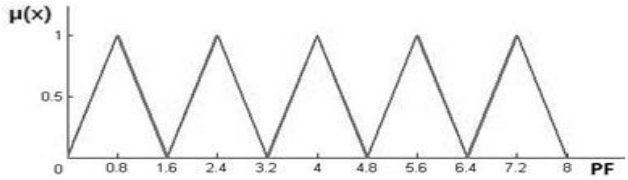


Figura 19. Conjuntos borrosos propuestos para la evaluación del peso de los PF en una EE

<u>Salida externa (SE)</u>
TED Muy Bajo: Trapecio (x, nulo, 1, 3, 7)
TED Bajo: Trapecio (x, 1, 5, 7, 9)
TED Medio: Trapecio (x, 6, 9, 16, 20)
TED Alto: Trapecio (x, 10, 16, 21, 28)
TED Muy Alto: Trapecio (x, 21, 28, nulo, nulo)

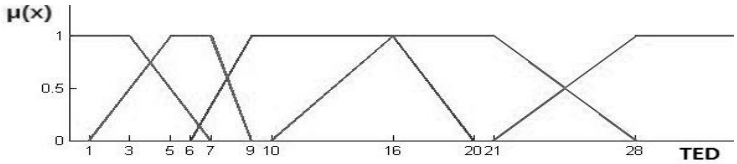


Figura 20. Conjuntos borrosos propuestos para la evaluación de la cantidad de TED en una SE

TAR Bajo: Trapecio (x, nulo, 0, 1, 2)
TAR Medio: Trapecio (x, 0, 2, 3, 4)
TAR Alto: Trapecio (x, 2, 5, nulo, nulo)

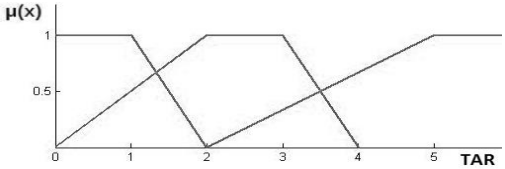


Figura 21. Conjuntos borrosos propuestos para la evaluación de la cantidad de TAR en una SE

PF Muy Bajo: Triángulo (x, 0, 0.9, 1.8)
PF Bajo: Triángulo (x, 1.8, 2.7, 3.6)
PF Medio: Triángulo (x, 3.6, 4.5, 5.4)
PF Alto: Triángulo (x, 5.4, 6.3, 7.2)
PF Muy Alto: Triángulo (x, 7.2, 8.1, 9)

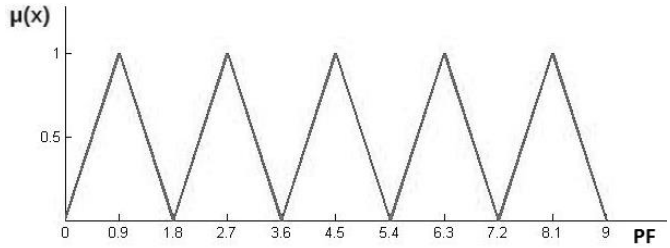


Figura 22. Conjuntos borrosos propuestos para la evaluación del peso de los PF en una SE

Consulta externa (CE)

TED Muy Bajo: Trapecio (x, nulo, 1, 3, 7)
TED Bajo: Trapecio (x, 1, 5, 7, 9)
TED Medio: Trapecio (x, 6, 9, 16, 20)
TED Alto: Trapecio (x, 10, 16, 21, 28)
TED Muy Alto: Trapecio (x, 21, 28, nulo, nulo)

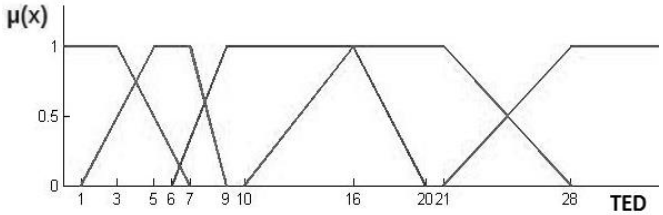


Figura 23. Conjuntos borrosos propuestos para la evaluación de la cantidad de TED en una CE

TAR Bajo: Trapecio (x, nulo, 0, 1, 2)
TAR Medio: Trapecio (x, 0, 2, 3, 4)
TAR Alto: Trapecio (x, 2, 5, nulo, nulo)

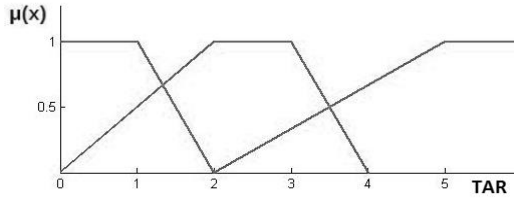


Figura 24. Conjuntos borrosos propuestos para la evaluación de la cantidad de TAR en una CE

PF Muy Bajo: Triángulo (x, 0, 0.8, 1.6)
PF Bajo: Triángulo (x, 1.6, 2.4, 3.2)
PF Medio: Triángulo (x, 3.2, 4, 4.8)
PF Alto: Triángulo (x, 4.8, 5.6, 6.4)

PF Muy Alto: Triángulo (x, 6.4, 7.2, 8)

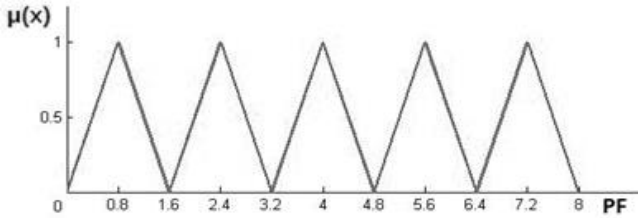


Figura 25. Conjuntos borrosos propuestos para la evaluación del peso de los PF en una CE

2.2.2. Paso 2: Definición de las reglas

Las reglas borrosas consisten en la combinación de uno o más conjuntos borrosos de entrada, a los cuales se les llama premisas o antecedentes, a las que se le asocia un conjunto borroso como salida, a la que se le llama conclusión o consecuente. La combinación de los conjuntos borrosos de entrada se realiza mediante expresiones lógicas, las cuales se traducen en una operación entre los combinados. Todas las reglas definidas forman parte entonces de la base de reglas que permite expresar el conocimiento de una forma completa. [Maguiña 2010]

Tabla 22: Memoria asociativa borrosa propuesta para el algoritmo

Base de Reglas para los PF		**TED**				
		Muy Bajo	**Bajo**	**Medio**	**Alto**	**Muy Alto**
TER / TAR	**Bajo**	Muy Bajo	Muy Bajo	Medio	Alto	Alto
	Medio	Muy Bajo	Bajo	Medio	Alto	Muy Alto
	Alto	Bajo	Bajo	Medio	Muy Alto	Muy Alto

Tabla 23: Relación de reglas resultantes para el algoritmo

Número	**TED**	**TER/TAR**	**PF**
1.	Muy Bajo	Bajo	Muy Bajo
2.	Muy Bajo	Medio	Muy Bajo
3.	Muy Bajo	Alto	Bajo
4.	Bajo	Bajo	Muy Bajo
5.	Bajo	Medio	Bajo
6.	Bajo	Alto	Bajo
7.	Medio	Bajo	Medio
8.	Medio	Medio	Medio
9.	Medio	Alto	Medio
10.	Alto	Bajo	Alto
11.	Alto	Medio	Alto
12.	Alto	Alto	Muy Alto
13.	Muy Alto	Bajo	Alto
14.	Muy Alto	Medio	Muy Alto
15.	Muy Alto	Alto	Muy Alto

La base de reglas se representa como una memoria asociativa borrosa (MAB), donde se define la consecuencia para cada combinación de dos entradas en una matriz. En la tabla 22 se muestra la MAB definida para el algoritmo.

Teniendo en cuenta la MAB definida se especifican 15 reglas a evaluar en el proceso de estimación utilizando el algoritmo propuesto. Se debe especificar que en todas las reglas es utilizada la expresión lógica “y”, lo cual deriva en la operación de intersección entre los conjuntos difusos de entrada, calculándose el mínimo en todos los casos. En la tabla 23 se muestran las reglas definidas para el sistema de inferencia borroso en el algoritmo propuesto.

2.2.3. Paso 3: Medición de los componentes del software

Para realizar la medición del software que se estima teniendo en cuenta la documentación generada hasta el momento sobre las funcionalidades que deben ser desarrolladas, se identifican los componentes AIL, AIE, EE, SE y CE del software a desarrollar. Luego se realiza el conteo de las funciones de datos y transaccionales de cada uno de los componentes que fueron identificados para obtener la cantidad de TED, TER y TAR de cada uno. Para realizar un conteo correcto de las funciones de datos y transaccionales en cada uno de los componentes deben ser aplicadas todas las aclaraciones, reglas, guías y criterios que se encuentran definidas en el manual de prácticas de conteo en [IFPUG 2010]. En la tabla 24 se muestra un ejemplo de los datos que se esperan obtener al finalizar este paso y que serán utilizados como ejemplos en las secciones siguientes del algoritmo.

Tabla 24: Conteo de las funciones de datos y transaccionales de un software

Componentes	Cantidad	TED	TER / TAR
Entradas Externas	2	4	2
	2	9	3
	2	19	4
Consultas Externas	1	4	2
	1	11	4
	1	21	5
Salidas Externas	3	5	1
	2	12	3
	2	19	4
Archivos Internos Lógicos	4	10	2
	2	20	4
	1	38	5
Archivos de Interfaz Externos	1	10	2
	1	20	2
	1	52	5

2.2.4. Paso 4: Determinación del grado de pertenencia

Para determinar el grado de pertenencia de cada uno de los valores obtenidos a la variable de entrada a la que se corresponde, se aplican las fórmulas definidas en el modelo matemático de las funciones trapezoidales que se muestra en la figura 10. En la tabla 25 y 26 se muestran los grados de pertenencia calculados para el ejemplo que se utiliza.

Tabla 25: Grados de pertenencia para la etiqueta lingüística TED

Variable de entrada	TED	$\mu_{TED\ MB}$	$\mu_{TED\ B}$	$\mu_{TED\ M}$	$\mu_{TED\ A}$	$\mu_{TED\ MA}$
Entradas Externas	4	0.75	0.75	0	0	0
	9	0	0.5	1	0	0
	19	0	0	0	0.75	0.5
Consultas Externas	4	0.75	0.75	0	0	0
	11	0	0	1	0.17	0
	21	0	0	0	1	0
Salidas Externas	5	0.5	1	0	0	0
	12	0	0	1	0.33	0
	19	0	0	0.25	1	0
Archivos Internos Lógicos	10	0	1	0	0	0
	20	0	0	1	0	0
	38	0	0	0.71	1	0
Archivos de Interfaz Externos	10	0	1	0	0	0
	20	0	0	1	0	0
	52	0	0	0.71	1	0

Tabla 26: Grados de pertenencia para las etiquetas lingüísticas TER y TAR

Variable de entrada	TER / TAR	$\mu_{TER/TAR\ B}$	$\mu_{TER/TAR\ M}$	$\mu_{TER/TAR\ A}$
Entrada Externa	2	0	1	0.5
	3	0	1	1
	4	0	0	1
Consulta Externa	2	0	1	0
	4	0	0	0.67
	5	0	0	1
Salida Externa	1	1	0.5	0
	3	0	1	0.33
	4	0	0	0.67
Archivos Internos Lógicos	2	1	0.5	0
	4	0	1	0.33
	5	0	1	0.67
Archivos de Interfaz Externos	2	1	0.5	0
	2	1	0.5	0
	5	0	1	0.67

2.2.5. Paso 5: Definición de los datos de salida

Teniendo en cuenta la MAB definida en el paso 2 del algoritmo, se aplica un operador T-normas para obtener el conjunto borroso de salida de cada regla en cada uno de los componentes medidos. Luego de obtener todos los conjuntos borrosos de salida para cada una de las reglas, se forma un único conjunto borroso de salida por cada componente, de forma tal que este pueda ser transformado en un valor nítido, a través de un proceso de concreción, para ser utilizado en el proceso de estimación del esfuerzo que se le debe dedicar al software que se desarrolla. Para obtener un único conjunto borroso de salida se aplica un operador T-conormas.

Luego del estudio realizado en esta investigación sobre los operadores T-normas y T-conormas que se han definido en la teoría de los conjuntos difusos, la T-norma elegida para esta investigación fue el mínimo y la T-conorma elegida fue el máximo. Las cuales tienen la siguiente representación: T_{min} (a, b) = min (a, b) = A ∩ B y S_{max} (a, b) = max (a, b) = A U B. En la tabla 27 se muestran los datos de salida para cada una de las reglas utilizando el operador T-norma del mínimo, así como en la tabla 28 se muestra el conjunto borroso de salida que se obtiene para cada componente al aplicar el operador T-conorma del máximo.

Tabla 27: Datos de salida para cada una de las reglas

Variable de entrada	TED	TER / TAR	R1	R2	R3	R4	R5	R6	R7	R8	R9	R11	R12	R15
Entrada Externa	4	2		0.75	0.5		0.75	0.5						
	9	3					0.5	0.5		1	1			
	19	4											0.75	0.5
Consulta Externa	4	2		0.75			0.75							
	11	4									0.67		0.17	
	21	5											1	
Salida Externa	5	1	0.5	0.5		1	0.5							
	12	3								1	0.33	0.33	0.33	
	19	4									0.25		0.67	
Archivos Internos Lógicos	10	2				1	0.5							
	20	4								1	0.33			
	38	5								0.71	0.67	1	0.67	
Archivos de Interfaz Externos	10	2				1	0.5							
	20	2							1	0.5				
	52	5								0.71	0.67	1	0.67	

Las reglas números 10, 13 y 14 no se muestran en la tabla 27 debido a que no se fueron utilizadas en ninguno de los componentes analizados para cada variable de entrada en el ejemplo de estimación que se desarrolla en los pasos del algoritmo.

Tabla 28: Conjunto borroso de salida para cada uno de los componentes

Variable de entrada	TED	TER / TAR	Salida
Entrada Externa	4	2	(0, 0.75, 0.5, 0, 0.75, 0.5, 0, 0, 0, 0, 0, 0, 0, 0, 0)
	9	3	(0, 0, 0, 0, 0.5, 0.5, 0, 1, 1, 0, 0, 0, 0, 0, 0)
	19	4	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.75, 0, 0, 0.5)
Consulta Externa	4	2	(0, 0.75, 0, 0, 0.75, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
	11	4	(0, 0, 0, 0, 0, 0, 0, 0, 0.67, 0, 0, 0.17, 0, 0, 0)
	21	5	(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0)
Salida Externa	5	1	(0.5, 0.5, 0, 1, 0.5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
	12	3	(0, 0, 0, 0, 0, 0, 0, 1, 0.33, 0, 0.33, 0.33, 0, 0, 0)
	19	4	(0, 0, 0, 0, 0, 0, 0, 0,0.25, 0, 0, 0.67, 0, 0, 0)
Archivos Internos Lógicos	10	2	(0, 0, 0, 1, 0.5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
	20	4	(0, 0, 0, 0, 0, 0, 0, 1, 0.33, 0, 0, 0, 0, 0, 0)
	38	5	(0, 0, 0, 0, 0, 0, 0, 0.71, 0.67, 0, 1, 0.67, 0, 0, 0)
Archivos de Interfaz Externos	10	2	(0, 0, 0, 1, 0.5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
	20	2	(0, 0, 0, 0, 0, 0, 1, 0.5, 0, 0, 0, 0, 0, 0, 0)
	52	5	(0, 0, 0, 0, 0, 0, 0, 0.71, 0.67, 0, 1, 0.67, 0, 0, 0)

2.2.6. Paso 6: Transformación del conjunto borroso de salida en un valor nítido

Luego para transformar el conjunto borroso de salida obtenido en un valor nítido, el modelo de Mamdani propone los métodos: Centro de área, Bisector del área, Media de máximo, Máximo más chico y Máximo más grande. Luego del estudio realizado en el capítulo 1 sobre estos métodos, para el proceso de concreción en esta investigación se propuso utilizar el método Centro de área.

Con la fórmula 20 se realiza el cálculo de los puntos de función sin ajustar para el primer componente, una entrada externa con 4 TED y 2 TAR, para luego mostrar en la tabla 29 los resultados del proceso de concreción de los datos en cada uno de los componentes, utilizando el método Centro de área. En la columna PFSA SIB Total, se muestra el resultado de multiplicar la cantidad de componentes que tienen la misma cantidad de TED y TER/TAR, con la cantidad de puntos de función que fueron calculados por el algoritmo que se le deben asignar a un componente con estas características. Además se muestran los puntos de función sin ajustar que le serían asignados a cada uno de los componentes al utilizar el método Puntos de Función en su forma tradicional. Al estimarse un software con las características del ejemplo que ha sido utilizado en los pasos del algoritmo propuesto, entonces, sus puntos de función sin ajustar se encuentran dados por los cálculos que se realizan mediante la fórmula 21.

$$PF_{EE}\,(4,2) = \frac{0.75 * 0.8 + 0.5 * 2.4 + 0.75 * 2.4 + 0.5 * 2.4}{0.75 + 0.5 + 0.75 + 0.5} = 1.92 \qquad (20)$$

Tabla 29: Puntos de función sin ajustar para cada uno de los componentes

Componentes	Cantidad	TED	TER / TAR	PFSA SIB	PFSA SIB Total	PFSA tradicional
Entrada Externa	2	4	2	1.92	3.84	6
	2	9	3	4	8	12
	2	19	4	6.56	13.12	12
Consulta Externa	1	4	2	1.6	1.6	3
	1	11	4	4.64	4.64	6
	1	21	5	7.2	7.2	6
Salida Externa	3	5	1	1.26	3.78	12
	2	12	3	5.57	11.14	10
	2	19	4	8.1	16.2	14
Archivos Internos Lógicos	4	10	2	3	12	28
	2	20	4	9	18	20
	1	38	5	11.76	11.76	10
Archivos de Interfaz Externos	1	10	2	2.17	2.17	5
	1	20	2	6.5	6.5	7
	1	52	5	11.45	11.45	7

$$\text{PFSA} = \text{PFSA}_1 + \text{PFSA}_2 + \ldots + \text{PFSA}_{15} = 3.84 + 8 + \ldots + 11.45 = 131.40 \quad (21)$$

Al utilizarse el método tradicional de Puntos de Función, para el mismo escenario calculado anteriormente, se obtienen 158 puntos de función sin ajustar. Con estos valores se obtiene entonces una diferencia de 26.60 puntos de función entre los métodos de estimación. Esta cantidad representa una diferencia de 306 horas si se toma la tendencia que existe de asignar 11.50 horas por cada punto de función, lo cual representa una diferencia de ocho semanas para el proceso de desarrollo de software.

2.2.7. Paso 7: Determinación del factor de ajuste

Para determinar el factor de ajuste a utilizar en el cálculo de los puntos de función del sistema al que se le estima su tamaño, se utiliza la misma fórmula empleada por el método tradicional de Puntos de Función. En la fórmula 22 se muestra el cálculo que se realiza para determinar el factor de ajuste considerando la valoración realizada a las catorce características generales del sistema que se muestra en la tabla 29.

$$\text{FA} = 0.65 + 0.01 * \sum_{i=1}^{14} c_i = 0.65 + 0.01 * 33 = 0.98 \quad (22)$$

Tabla 30: Valoración de las características generales del sistema

Número	Característica	Valoración
1.	Comunicación de datos	2
2.	Procesamiento distribuido de datos	1
3.	Rendimiento	4
4.	Configuración altamente utilizada	3
5.	Promedio de transacciones	3
6.	Entrada de datos en línea	1
7.	Eficiencia para el usuario final	3
8.	Actualización en línea	1
9.	Procesamiento complejo	2
10.	Reusabilidad	3
11.	Facilidad de instalación	2
12.	Facilidad de operación	3
13.	Múltiples localizaciones	0
14.	Facilidad de cambios	5

2.2.8. Paso 8: Determinación de la cantidad de puntos de función del software

Luego de calculados los puntos de función sin ajustar en el paso seis y el factor de ajuste en el paso siete, solo queda realizar la multiplicación de estos dos valores para obtener los puntos de función a utilizar en la estimación del tamaño del software a desarrollar. En la fórmula 23 se muestra el cálculo que se realiza para determinar la cantidad de puntos de función del software que se estima.

$$\mathrm{PF} = 131.40 * 0.98 = 128.77 \quad (23)$$

2.2.9. Paso 9: Estimación del esfuerzo

Para estimar el esfuerzo que se le debe dedicar al desarrollo del software analizado como ejemplo en el algoritmo, se utilizó el valor medio de la industria, definido por el ISBSG, de 11.50 horas-hombre por punto de función. En la fórmula 24 se muestra el cálculo que se realiza para estimar las horas que se le deben dedicar al desarrollo de este sistema.

$$\mathrm{Esfuerzo} = 128.77 * 11.50 = 1\,481 \text{ horas} \quad (24)$$

2.2.10. Calibración de los números difusos de los datos de cada una de las reglas

Para calibrar un sistema de inferencia borroso se deben modificar los parámetros utilizados con el objetivo de obtener un mayor grado de fidelidad de la realidad deseada. Para esto se pueden realizar modificaciones tanto a los números difusos de los datos de entradas de cada regla como a los datos de salida en cada regla definida. En los casos donde se utiliza el criterio de expertos o los datos

históricos se recomienda realizar modificaciones solamente a los datos de salida de las reglas. Se debe tener en cuenta que al finalizar la fase de calibración la solución final obtenida no debe ser muy diferente a la definida inicialmente.

En esta investigación para calibrar los números difusos utilizados en la propuesta se utiliza la técnica de prueba y error de forma tal que en los pesos obtenidos con el método no presenten límites bruscos. En el capítulo 3 de esta investigación se muestra el funcionamiento del método propuesto a través de su aplicación en tres casos de estudio. Se recomienda que los escenarios escogidos para la validación de la propuesta sean diferentes a los escenarios que hayan sido utilizados para la calibración de la propuesta.

2.3. Conclusiones parciales

Con la elaboración en este capítulo de dos algoritmos para estimar el esfuerzo en un proyecto de desarrollo de software a partir de puntos de función, se alcanzaron a las siguientes conclusiones:

- ✓ Se lograron aplicar las técnicas de *soft computing* al método Puntos Función para la estimación del esfuerzo de un proyecto de desarrollo de software utilizando la lógica difusa y la Computación con Palabras, lo que permite dar cumplimiento al segundo objetivo específico planteado en la investigación.
- ✓ Se establecieron dos algoritmos para estimar el esfuerzo en un proyecto de desarrollo de software por puntos de función, utilizando en el primero el modelo de representación lingüística de 2-tuplas y en el segundo un Sistema de Inferencia Borrosa del tipo Mamdani, lo que permite dar cumplimiento al tercer objetivo específico planteado en la investigación.
- ✓ Para utilizar el modelo de representación lingüística de 2-tuplas en el algoritmo fue necesario modificar uno de los coeficientes utilizados en el ajuste de la complejidad técnica del sistema para determinar el factor de ajuste a utilizar en el cálculo de los puntos de función ajustados.

CAPÍTULO 3: APLICACIÓN DEL ALGORITMO Y ANÁLISIS DE RESULTADOS

En este capítulo se validan los algoritmos propuestos en la investigación para estimar el esfuerzo en un proyecto de desarrollo de software a partir de puntos de función y técnicas de *soft computing*, a través de los resultados que se alcanzaron al aplicar los dos algoritmos en tres casos de estudio. Los proyectos de desarrollo de software escogidos como casos de estudio pertenecen al centro CEIGE de la Facultad 3. Además de realizarse una valoración del impacto económico y social de la propuesta.

3.1. Aplicación de la propuesta en el caso de estudio Depósitos de Aduana

El sistema Depósitos de Aduana, gestiona el control del inventario de cada depósito de la Aduana General de la República de Cuba (AGR), así como el control en el destino final de las mercancías almacenadas en el mismo y la emisión de un conjunto de alertas sobre violaciones a la normativa aduanera cometidas en estos. De forma tal que se permite agilizar la entrega y procesamiento de la información de los documentos para el control de la entrada, movimientos y salidas de las mercancías en los depósitos, proporcionando un mecanismo de configuración para la gestión de los tiempos establecidos en las resoluciones aduanales. Además este sistema proporciona un conjunto de reportes que permita llevar el control exacto de los inventarios de los depósitos de la AGR y de los documentos emitidos por cada depósito.

A este sistema le fueron desarrolladas 39 funcionalidades; las mismas fueron clasificadas por su complejidad de la siguiente manera: 10 en Alta, 15 en Media y 14 en Baja. Teniendo en cuenta estos datos y según el método de estimación utilizado en el proyecto, que es el método definido por la UCI para todos los proyectos de desarrollo de software de la universidad, el esfuerzo estimado para el desarrollo de este sistema fue de 2 900 horas.

A continuación se describe la aplicación del método Puntos de Función tradicional y los dos algoritmos propuestos en esta investigación, al sistema Depósitos de Aduana. En la tabla 31 se muestra el conteo de las funciones de datos y transaccionales realizado al sistema Depósitos de Aduana siguiendo la metodología definida para el método Puntos de Función en [Ifpug 2010]. Además en la tabla 32 se muestra la valoración de las catorce características generales para el sistema seleccionado como caso de estudio.

Con la valoración de las características generales del sistema realizada se determinó que el factor de ajuste para este sistema estaría dado por 0.67. Para estimar la cantidad de horas necesarias para el desarrollo de este sistema, se utilizó el valor medio de la industria 11.50 horas-hombre por punto de función; esta media general fue definida por el ISBSG. En la tabla 33 se muestran los resultados que se obtuvieron al estimar el sistema Depósitos de Aduana con el método Puntos de Función tradicional y los dos algoritmos propuestos en esta investigación; donde el tiempo de desarrollo que se obtiene se corresponde al tiempo que debe ser dedicado por una sola persona para desarrollar en su totalidad el sistema.

Tabla 31: Conteo de las funciones de datos y transaccionales del sistema Depósitos de Aduana

Componentes	Cantidad	TED	TER / TAR	Componentes	Cantidad	TED	TER / TAR
Entradas Externas	1	7	3	Archivos Internos Lógicos	1	24	7
	1	23	7		1	25	5
	1	24	5		1	31	7
	1	41	10		1	42	10
	1	50	9		1	51	9
	1	68	10		1	69	10
	1	75	12		1	76	12
Consultas Externas	2	14	3	Salidas Externas	1	14	3
	2	15	3		1	15	2
	2	16	3		1	18	3
	1	16	4		1	19	2
	1	16	5		1	20	3
	1	19	3		1	21	3
	1	28	8		1	21	2
	1	29	8		1	21	4
	1	34	8		1	23	4
	1	35	8		1	25	3
Salidas Externas	1	5	2		2	27	3
	2	11	1		1	29	4
	1	13	2		1	36	6

Tabla 32: Características generales del sistema Depósitos de Aduana

Número	Característica	Valoración
1.	Comunicación de datos	Medio
2.	Procesamiento distribuido de datos	Sin Influencia
3.	Rendimiento	Significativo
4.	Configuración altamente utilizada	Medio
5.	Promedio de transacciones	Medio
6.	Entrada de datos en línea	Medio
7.	Eficiencia para el usuario final	Significativo
8.	Actualización en línea	Sin Influencia
9.	Procesamiento complejo	Accidental
10.	Reusabilidad	Medio
11.	Facilidad de instalación	Moderado
12.	Facilidad de operación	Significativo
13.	Múltiples localizaciones	Sin Influencia
14.	Facilidad de cambios	Moderado

Tabla 33: Resultados de la estimación realizada al sistema Depósitos de Aduana

Método	Puntos de Función sin Ajustar	Puntos de Función	Tiempo de desarrollo (h)
Puntos de Función tradicional	313	210	2 415
Algoritmo utilizando 2-tuplas	328.51	220	2 530
Algoritmo utilizando SIB	402.04	269	3 094

Para determinar la precisión de la estimación obtenida, se compararon las horas estimadas para su desarrollo en cada uno de los métodos contra las horas reales invertidas en su desarrollo. Según los datos recogidos en el expediente del proyecto, fueron necesarias emplear para su desarrollo 400 horas con un equipo de trabajo integrado por 6 personas. Esta información nos permite asumir que si hubiera trabajado en el desarrollo de este sistema una sola persona, entonces la cantidad de horas dedicadas al mismo hubieran sido de 2 400. En la figura 26 se muestra la diferencia de horas que tuvieron los cuatro métodos de estimación respecto a las horas que fueron invertidas en el desarrollo.

Como se puede observar en la figura 26, la estimación más cercana al esfuerzo real que le fue dedicado al desarrollo del sistema Depósitos de Aduana fue el método Puntos de Función tradicional, que estimó con una diferencia del 1 % por encima solamente. En el caso del algoritmo propuesto con 2-tuplas se obtuvo una diferencia del 5 % por encima, así como una diferencia del 21 % por encima con el método de estimación definido en la UCI. Los peores resultados estuvieron asociados al algoritmo propuesto con un SIB, el cual obtuvo una diferencia del 29 % por encima de lo sucedido.

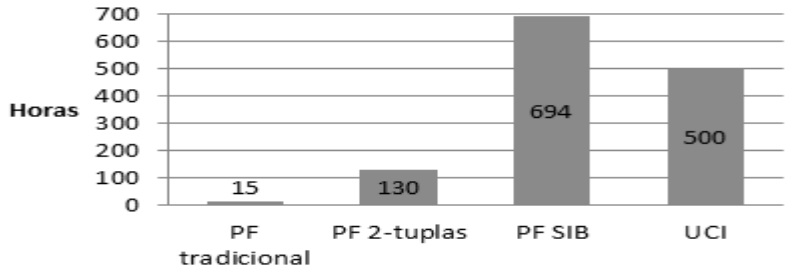


Figura 26. Variación de las horas estimadas en cada método respecto a las horas invertidas en el desarrollo del sistema Depósitos de Aduana

3.2. Aplicación de la propuesta en el caso de estudio Despacho Comercial

El sistema Despacho Comercial, gestiona el control de las operaciones de importación y exportación con carácter comercial en la Aduana General de la República de Cuba (AGR), así como la recepción y procesamiento de la información electrónica de los documentos del despacho comercial. De forma tal que se permite agilizar la entrega y procesamiento de la información de los documentos para el despacho comercial de las mercancías que entran a salen del país. Se debe señalar que los datos tomados para el caso de estudio de este software sólo pertenecen al desarrollo de los módulos Solicitudes y Documentos Complementarios.

A este sistema le fueron desarrolladas 59 funcionalidades; las mismas fueron clasificadas por su complejidad de la siguiente manera: 29 en Alta, 12 en Media y 18 en Baja. Teniendo en cuenta estos datos y según el método de estimación utilizado en el proyecto, que es el método definido por la UCI para todos los proyectos de desarrollo de software de la universidad, el esfuerzo estimado para el desarrollo de este sistema era de 4 388 horas.

A continuación se describe la aplicación del método Puntos de Función tradicional y los dos algoritmos propuestos en esta investigación, al sistema Despacho Comercial. En la tabla 34 se muestra el conteo de las funciones de datos y transaccionales realizado al sistema Despacho Comercial siguiendo la metodología definida para el método Puntos de Función en [Ifpug 2010]. Además en la tabla 35 se muestra la valoración de las catorce características generales para el sistema seleccionado como caso de estudio.

Tabla 34: Conteo de las funciones de datos y transaccionales del sistema Despacho Comercial

Componentes	**Cantidad**	**TED**	**TER / TAR**	**Componentes**	**Cantidad**	**TED**	**TER / TAR**
Entradas Externas	4	9	3	Archivos Internos Lógicos	4	10	3
	2	11	3		2	12	3
	1	12	3		1	13	3
	2	14	3		2	15	3
	2	15	3		2	16	3
	2	16	4		2	17	4
	2	17	4		2	18	4
	1	18	3		1	19	3
	1	18	4		1	19	4
	2	19	3		2	20	3
	1	20	4		1	21	3
	1	21	3		1	22	3
	1	21	4		1	22	4
	2	22	3		2	23	3
	1	23	3		1	24	3
	2	26	3		2	27	3
	1	37	3		1	38	3
Salidas Externas	1	5	1	Consultas Externas	1	9	1
	1	23	3	Salidas Externas	1	33	3
	2	26	4		1	33	4
	1	27	3		1	33	5
	2	28	3		1	35	4
	1	29	3		1	37	5
	1	32	3		2	38	5

Tabla 35: Características generales del sistema Despacho Comercial

Número	Característica	Valoración
1.	Comunicación de datos	Medio
2.	Procesamiento distribuido de datos	Sin Influencia
3.	Rendimiento	Significativo
4.	Configuración altamente utilizada	Medio
5.	Promedio de transacciones	Medio
6.	Entrada de datos en línea	Medio
7.	Eficiencia para el usuario final	Significativo
8.	Actualización en línea	Sin Influencia
9.	Procesamiento complejo	Accidental
10.	Reusabilidad	Medio
11.	Facilidad de instalación	Moderado
12.	Facilidad de operación	Significativo
13.	Múltiples localizaciones	Sin Influencia
14.	Facilidad de cambios	Moderado

Con la valoración de las características generales del sistema realizada se determinó que el factor de ajuste para este sistema estaría dado por 0.67. Para estimar la cantidad de horas necesarias para el desarrollo de este sistema, se utilizó el valor medio de la industria 11.50 horas-hombre por punto de función; esta media general fue definida por el ISBSG. En la tabla 36 se muestran los resultados que se obtuvieron al estimar el sistema Despacho Comercial con el método Puntos de Función tradicional y los dos algoritmos propuestos en esta investigación; donde el tiempo de desarrollo que se obtiene se corresponde al tiempo que debe ser dedicado por una persona para el desarrollo del sistema.

Para determinar la precisión de la estimación obtenida por cada uno de los métodos analizados, se comparó las horas estimadas para su desarrollo en cada uno de los métodos contra las horas reales invertidas en el desarrollo del sistema. Según los datos recogidos en el expediente del proyecto para el desarrollo del sistema fueron necesarias emplear 520 horas por parte del equipo de desarrollo, el cual estuvo integrado por 8 personas. Esta información nos permite asumir que si hubiera trabajado en el desarrollo de este sistema una sola persona, entonces la cantidad de horas dedicadas al mismo hubieran sido de 4 160. En la figura 27 se muestra la diferencia de horas que tuvieron los cuatro métodos de estimación analizados en esta investigación respecto a las horas que fueron invertidas en el desarrollo del sistema Despacho Comercial.

Tabla 36: Resultados de la estimación realizada al sistema Despacho Comercial

Método	Puntos de Función sin Ajustar	Puntos de Función	Tiempo de desarrollo (h)
Puntos de Función tradicional	509	341	3 922
Algoritmo utilizando 2-tuplas	543.51	364	4 186
Algoritmo utilizando SIB	490.78	329	3 784

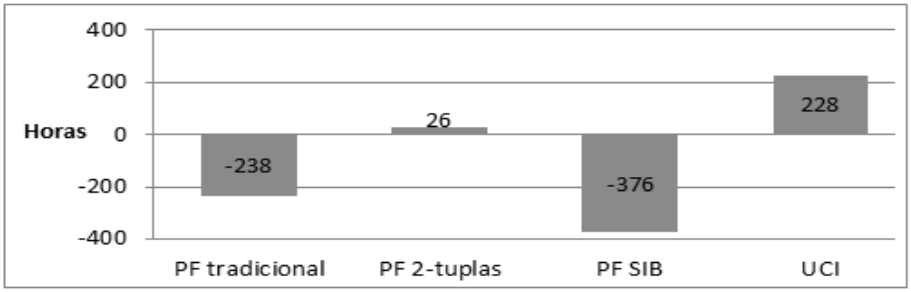


Figura 27. Variación de las horas estimadas en cada método respecto a las horas invertidas en el desarrollo del sistema Despacho Comercial

Como se puede observar en la figura 27, la estimación más cercana al esfuerzo real que le fue dedicado al desarrollo del sistema Despacho Comercial fue con el algoritmo propuesto que utiliza el modelo de representación lingüística de 2-tuplas, que estimó con una diferencia del 1 % por encima solamente. En el caso del método de estimación definido en la UCI se obtuvo una diferencia del 5 % por encima, así como una diferencia del 6 % por debajo con el método Puntos de Función tradicional. Los peores resultados estuvieron asociados al algoritmo propuesto que utiliza un SIB, con el cual se obtuvo una diferencia del 9 % por debajo de lo que realmente sucedió.

3.3. Aplicación de la propuesta en el caso de estudio Control de MTI

El sistema Control de MTI, gestiona el control que se realiza la Aduana General de la República de Cuba (AGR) a los Medios de Transporte Internacional (MTI) en sus arribos, permanencias y salidas del país, así como el control de las medidas aplicadas a estos MTI en los controles que le fueron realizados. De forma tal que permite brindar informaciones precisas sobre las situaciones detectadas en el arribo, estancia o salida de un MTI al país. Además este sistema proporciona un conjunto de reportes que permita llevar el control exacto de los inventarios de los depósitos de la AGR y de los documentos emitidos por cada depósito.

A este sistema le fueron desarrolladas 21 funcionalidades; las mismas fueron clasificadas por su complejidad de la siguiente manera: 13 en Alta, 3 en Media y 5 en Baja. Teniendo en cuenta estos datos y según el método de estimación utilizado en el proyecto, que es el método definido por la UCI para todos los proyectos de desarrollo de software de la universidad, el esfuerzo estimado para el desarrollo de este sistema era de 1 562 horas.

A continuación se describe la aplicación del método Puntos de Función tradicional y los dos algoritmos propuestos en esta investigación, al sistema Control de MTI. En la tabla 37 se muestra el conteo de las funciones de datos y transaccionales realizado al sistema Control de MTI siguiendo la metodología definida para el método Puntos de Función en [Ifpug 2010]. Además en la tabla 38 se muestra la valoración de las catorce características generales para el sistema seleccionado como caso de estudio.

Tabla 37: Conteo de las funciones de datos y transaccionales del sistema Control de MTI

Componentes	Cantidad	TED	TER / TAR	Componentes	Cantidad	TED	TER / TAR
Entradas Externas	1	3	1	Archivos Internos Lógicos	1	3	2
	1	6	2		1	6	3
	1	9	1		1	9	2
	1	13	2		1	13	3
	2	15	2		1	14	4
	2	16	2		2	15	3
	1	19	3		2	21	2
	2	21	1		1	21	3
	1	21	2		1	22	2
	1	40	4		1	29	3
Consultas Externas	1	11	1		1	39	5
	1	12	1	Consultas Externas	1	14	2
	3	14	1		1	16	1

Tabla 38: Características generales del sistema Control de MTI

Número	Característica	Valoración
1.	Comunicación de datos	Medio
2.	Procesamiento distribuido de datos	Sin Influencia
3.	Rendimiento	Significativo
4.	Configuración altamente utilizada	Medio
5.	Promedio de transacciones	Medio
6.	Entrada de datos en línea	Medio
7.	Eficiencia para el usuario final	Significativo
8.	Actualización en línea	Sin Influencia
9.	Procesamiento complejo	Accidental
10.	Reusabilidad	Medio
11.	Facilidad de instalación	Moderado
12.	Facilidad de operación	Significativo
13.	Múltiples localizaciones	Sin Influencia
14.	Facilidad de cambios	Moderado

Tabla 39: Resultados de la estimación realizada al sistema Control de MTI

Método	Puntos de Función sin Ajustar	Puntos de Función	Tiempo de desarrollo (h)
Puntos de Función tradicional	195	131	1 507
Algoritmo utilizando 2-tuplas	218.3	146	1 679
Algoritmo utilizando SIB	184.74	124	1 426

Con la valoración de las características generales del sistema realizada se determinó que el factor de ajuste para este sistema estaría dado por 0.67. Para estimar la cantidad de horas necesarias para el

desarrollo de este sistema, se utilizó el valor medio de la industria 11.50 horas-hombre por punto de función; esta media general fue definida por el ISBSG. En la tabla 39 se muestran los resultados que se obtuvieron al estimar el sistema Control de MTI con el método Puntos de Función tradicional y los dos algoritmos propuestos en esta investigación; donde el tiempo de desarrollo que se obtiene se corresponde al tiempo que debe ser dedicado por una persona para el desarrollo del sistema.

Para determinar la precisión de la estimación obtenida por cada uno de los métodos analizados, se comparó las horas estimadas para su desarrollo en cada uno de los métodos contra las horas reales invertidas en el desarrollo del sistema. Según los datos recogidos en el expediente del proyecto para el desarrollo del sistema fueron necesarias emplear 232 horas por parte del equipo de desarrollo, el cual estuvo integrado por 7 personas. Esta información nos permite asumir que si hubiera trabajado en el desarrollo de este sistema una sola persona, entonces la cantidad de horas dedicadas al mismo hubieran sido de 1 624. En la figura 28 se muestra la diferencia de horas que tuvieron los cuatro métodos de estimación analizados en esta investigación respecto a las horas que fueron invertidas en el desarrollo del sistema Control de MTI.

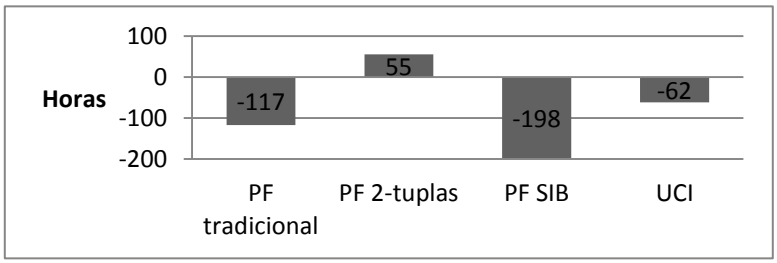


Figura 28. Variación de las horas estimadas en cada método respecto a las horas invertidas en el desarrollo del sistema Control de MTI

Como se puede observar en la figura 28, la estimación más cercana al esfuerzo real que le fue dedicado al desarrollo del sistema Control de MTI fue con el algoritmo que utiliza el modelo de representación lingüística de 2-tuplas, que estimó con una diferencia del 3 % por encima. En el caso del método de estimación definido en la UCI se obtuvo una diferencia del 4 % por debajo, así como una diferencia del 7 % por debajo con el método Puntos de Función tradicional. Los peores resultados estuvieron asociados nuevamente al algoritmo que utiliza un SIB, con el cual se obtuvo una diferencia del 12 % por debajo de lo que realmente sucedió.

3.4. Análisis de los resultados de la aplicación de los algoritmos en los casos de estudio

En la figura 29 son mostradas las horas que fueron invertidas en el desarrollo de los tres sistemas analizados en las secciones anteriores, junto a las horas que fueron estimadas por cada uno de los métodos de estimación analizados en la investigación. Teniendo en cuenta estos datos se considera que el método con estimaciones más precisas fue el algoritmo propuesto que utiliza 2-tuplas; las estimaciones menos precisas fueron obtenidas con la aplicación del algoritmo propuesto que utiliza un SIB.

Las variaciones de horas obtenidas en cada uno de los métodos de estimación analizados, salvo con el algoritmo propuesto que utiliza un SIB y el método establecido en la UCI en el proyecto Depósitos de Aduana, estuvieron siempre en un rango de hasta un 10 % por encima o por debajo de las horas dedicadas al desarrollo. En las dos excepciones se obtuvieron una diferencia de un 29 % y un 21 % por encima de lo ocurrido en el proyecto de desarrollo de software.

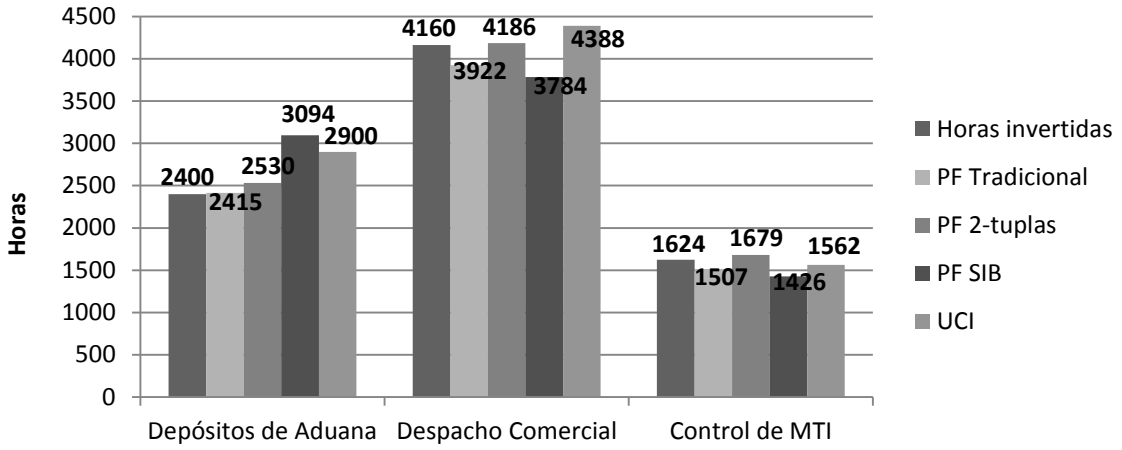


Figura 29. Horas invertidas en el desarrollo de los sistemas de los casos de estudios y las horas estimadas por cada método de estimación analizado

El algoritmo que utiliza un SIB requiere la utilización de los registros históricos disponibles sobre desarrollos de software previos para determinar los conjuntos borrosos de las variables de entrada para realizar estimaciones más precisas. Se debe señalar que la consideración anterior se encuentra en correspondencia con el paso de la propuesta del algoritmo donde se realiza la calibración de los números difusos del SIB. Se recomienda también analizar la utilización de operadores T-normas y T-conormas para la fusión de los datos que sean diferentes a los utilizados.

Para lograr un correcto funcionamiento de los dos algoritmos propuestos se considera necesario garantizar una homogeneidad en los artefactos generados en el proceso de desarrollo de software que serán utilizados en el conteo de las funciones de datos y transaccionales. En caso de no existir una homogeneidad en los artefactos utilizados para el conteo, se comprometerá la precisión de la estimación del tamaño al no emplearse las cantidades reales de TED, TER y TAR en cada uno de los componentes medidos.

Por último se considera que un factor fundamental para garantizar el correcto funcionamiento de los dos algoritmos propuestos en la investigación, lo constituye el conteo de las funciones de datos y transaccionales del software que se estima. Para esto deben ser estudiadas con profundidad todas las aclaraciones, reglas, guías y criterios que contiene el manual de prácticas de conteo de la IFPUG. Al garantizarse este factor se podrán ir creando registros históricos consistentes con todos los sistemas que se estimen al realizarse un conteo correcto y homogéneo, lo que permitirá calibrar los conjuntos utilizados en cada una de las propuestas para lograr una mayor precisión en la estimación.

3.5. Análisis del impacto económico y social de la implantación de la propuesta

La implantación del algoritmo propuesto, ya sea utilizando un SIB o el modelo de representación lingüística de 2-tuplas, requiere de un conjunto de acciones que permitan obtener un resultado exitoso. Para estimar el costo de implantación de la propuesta en la UCI, se asumirá que se realizará la implantación en al menos un proyecto de cada uno de los centros de desarrollo que tiene esta universidad. De acuerdo a estos datos se muestra en tabla 40 el comportamiento del costo que tendría la implantación de la propuesta en un proyecto de un centro de desarrollo, siguiendo el conjunto de acciones que se especifican.

Se debe destacar que para su implantación no se requiere de una gran cantidad de recursos económicos (computadoras, hojas y bolígrafos), siendo el tiempo el recurso determinante para calcular el costo de implantación. En la estimación se utilizó una tarifa horaria de $ 5.97 por hora de trabajo de cada persona involucrada. Esta tarifa horaria responde al salario más alto de las personas involucradas, en este caso el de un Jefe de Departamento con categoría docente de Instructor y con una evaluación de Adecuado en su desempeño trimestral.

En la etapa de planificación se propone que participe la persona capacitada para realizar la implantación de la propuesta y el Jefe del Proyecto donde se va a implantar, donde se deben revisar los aseguramientos para garantizar el éxito del proceso. Los directivos a participar en las actividades son: Jefe de Departamento, Jefe de Proyecto y Analista Principal del proyecto. En la capacitación teórica, la capacitación práctica y el análisis de resultados, deben participar además de los directivos, el personal involucrado a tareas de planificación y análisis que se consideren necesarios.

Tabla 40: Estimación del costo de aplicación de la propuesta en un proyecto de un centro de desarrollo

Acción	Tiempo (h)	Personal	Costo ($)
Planificación para la implantación	8	2	95.52
Encuentro inicial con directivos	2	4	47.76
Capacitación teórica	4	8	191.04
Capacitación práctica	4	8	191.04
Análisis de los resultados	4	8	191.04
Informe de los resultados	2	4	47.76
Costo total para un proyecto			764.16

Con este análisis de puede concluir que el costo de implantación de la propuesta en la UCI es relativamente bajo respecto al impacto que puede tener la aplicación de la propuesta en los proyectos de desarrollo. Se destaca también que se estimó con el salario más alto posible para el personal involucrado e incorporando cuatro personas a capacitar en cada proyecto, números que pueden ser menores si la dirección del centro de desarrollo donde se implante lo considera pertinente. Con la incorporación de un módulo a la herramienta GESPRO que permita el cálculo de los puntos de función siguiendo una o las dos generalizaciones detalladas en esta investigación, se minimiza el esfuerzo y el tiempo que se le dedica a la aplicación de la propuesta.

Para incluir este módulo a la herramienta, contando con la experiencia de los especialistas del Laboratorio de Investigaciones de Gestión de Proyecto que mantienen esta herramienta, se estima que el costo de desarrollo sea como se muestra en la tabla 41. Este costo de desarrollo se considera relativamente bajo también y los beneficios que se reportarán serán mayores, al poder realizar en todos los centros de desarrollo de la universidad la estimación utilizando el algoritmo propuesto a través de la herramienta. En la estimación se utilizó una tarifa horaria de $ 4.15 por hora de trabajo de cada persona involucrada, teniendo en cuenta el salario que percibe un especialista de un centro de desarrollo de la UCI. De igual manera que en la implantación de la propuesta, el desarrollo de este módulo no requiere de una gran cantidad de recursos económicos (computadoras, hojas y bolígrafos), siendo el tiempo el recurso determinante para calcular el costo de desarrollo.

Tabla 41: Estimación del costo de desarrollo de la propuesta para la herramienta GESPRO

Fases	**Tiempo (h)**	**Personal**	**Costo ($)**
Análisis y diseño	8	1	33.20
Desarrollo	24	2	199.20
Pruebas	8	1	33.20
Costo de desarrollo			265.60

Con este análisis de puede concluir que el costo de desarrollo de la propuesta para su inclusión en la herramienta GESPRO es relativamente bajo respecto al impacto que puede tener la aplicación de la propuesta en los proyectos de desarrollo de la UCI. A partir de las estimaciones realizadas se obtiene como costo para la utilización en la UCI, de uno de los dos algoritmos propuestos en esta investigación, un total de 1 029.76 pesos.

La utilización de uno de los algoritmos propuestos en esta investigación, a través de la herramienta o de forma manual, contribuirá a obtener una estimación más precisa del tamaño del software a desarrollar. Esto permitirá realizar una planificación más exacta del proceso de desarrollo en el proyecto, lo que contribuirá a utilizar con mayor eficiencia los recursos del proyecto al planificar con mayor exactitud sus necesidades dentro del proceso de desarrollo. Además de proporcionar una información más precisa para calcular los costos asociados al desarrollo del software, lo que ayudará a la alta gerencia del proyecto a tomar las decisiones que se necesiten en cada momento.

3.5.1. Lineamiento de la Política Económica y Social del Partido y la Revolución

Las propuestas realizadas en la presente investigación se encuentran en correspondencia con los Lineamientos de la Política Económica y Social del Partido y la Revolución aprobados en el VI Congreso del Partido Comunista de Cuba para actualizar el modelo económico cubano garantizando la continuidad del Socialismo [PCC 2011]. Los lineamientos que se apoyan con las propuestas realizadas en la investigación se muestran a continuación.

- ✓ Lineamiento 7. Lograr que el sistema empresarial del país esté constituido por empresas eficientes, bien organizadas y eficaces, y serán creadas las nuevas organizaciones superiores

de dirección empresarial. Se desarrollará la cooperación entre las empresas para garantizar mayor eficiencia y calidad. Se elaborará la norma jurídica que regule todos estos aspectos.

- ✓ Lineamiento 131. Sostener y desarrollar los resultados alcanzados en el campo de la biotecnología, la producción médico-farmacéutica, la industria del software y el proceso de informatización de la sociedad, las ciencias básicas, las ciencias naturales, los estudios y el empleo de las fuentes de energía renovables, las tecnologías sociales y educativas, la transferencia tecnológica industrial, la producción de equipos de tecnología avanzada, la nanotecnología y los servicios científicos y tecnológicos de alto valor agregado.

Las propuestas realizadas en la presente investigación también se encuentran en correspondencia con el Decreto Número 327, "Reglamento del Proceso Inversionista", aprobado el 11 de octubre del 2014 por el Consejo de Ministros para regular los elementos esenciales del proceso inversionista en Cuba teniendo en cuenta la actualización del modelo económico del país [Consejo de Ministros 2015] y dentro de ella el artículo 28 que plantea lo siguiente:

- ✓ ARTÍCULO 28. El inversionista tiene las obligaciones y atribuciones siguientes: (…) 15. presentar, para su evaluación, el estudio de factibilidad técnico-económica de la inversión a la instancia que corresponda en los diferentes momentos evaluativos, y responder por la concepción de la inversión, por la calidad y precisión de los cálculos y estimaciones contenidas en dichos documentos; para ello, hace participar a los sujetos del proceso inversionista que sea necesario; (…)

3.6. Conclusiones parciales

Con la aplicación de los algoritmos propuestos en tres proyectos de desarrollo de software del centro CEIGE de la UCI y el análisis de los resultados obtenidos en cada uno, así como el análisis del impacto económico y social de la propuesta realizada en la investigación se concluye lo siguiente:

- ✓ La aplicación del algoritmo que utiliza 2-tuplas en los casos de estudio evidenció que es el método más preciso de los analizados para estimar el esfuerzo.
- ✓ Los resultados alcanzados con la aplicación del algoritmo que utiliza un SIB en los casos de estudio evidenció que se requiere la utilización de los registros históricos para la definición de los conjuntos borrosos de las variables de entrada.
- ✓ El costo de implantación de uno de los algoritmos se estima en 764.16 pesos, valor relativamente bajo, factible para su aplicación.
- ✓ El resultado estimado en 265.60 pesos para el desarrollo de un módulo en la herramienta GESPRO, permite valorar la implementación de los algoritmos propuestos.

CONCLUSIONES

Al término de la investigación se alcanzaron a las siguientes conclusiones:

- ✓ Para reducir los límites bruscos en los métodos de estimación del esfuerzo en el desarrollo de software, han sido utilizadas con éxito las técnicas de *soft computing*, donde el modelo de representación lingüística de 2-tuplas no había sido utilizado.

- ✓ Se elaboraron dos algoritmos para estimar el esfuerzo a partir de puntos de función y técnicas de *soft computing* basados en el modelo de representación lingüística de 2-tuplas y en un Sistema de Inferencia Borrosa del tipo Mamdani.

- ✓ Con la aplicación del algoritmo que utiliza el modelo de representación lingüística 2-tuplas se obtuvieron las estimaciones más precisas comparadas con el resto de los métodos, en los proyectos de desarrollo de software analizados como casos de estudio.

- ✓ El algoritmo propuesto que utiliza un SIB debe ser modificado para mejorar la precisión de sus estimaciones a partir de la utilización de los registros históricos para la definición de los conjuntos borrosos de las variables de entrada.

RECOMENDACIONES

A partir de los análisis y criterios que han sido expresados en la investigación se recomienda:

- ✓ Calibrar los conjuntos borrosos del algoritmo que utiliza un SIB, a partir de los registros históricos disponibles sobre desarrollos de software previos.
- ✓ Implementar un sistema que permita automatizar los algoritmos propuestos para facilitar su aplicación en los proyectos de desarrollo de software.
- ✓ Homogeneizar los artefactos generados en el proceso de desarrollo de software que serán utilizados en el conteo de las funciones de datos y transaccionales.

REFERENCIAS BIBLIOGRÁFICAS

ALBRECHT, Allan. “Measuring Application Development Productivity”. En: *Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium*. Monterey, California, Estados Unidos, Octubre de 1979. Disponible en: < http://www.bfpug.com.br/Artigos/Albrecht/MeasuringApplicationDevelopmentProductivity.pdf>.

BALLESTER, Luis; COLOM, Antonio J. “Lógica difusa una nueva epistemología para las Ciencias de la Educación”. *Revista de Educación*. 2006, no. 340, p. 995-1008. Madrid, España. ISSN: 0034-8082. Disponible en: <http://www.revistaeducacion.mec.es/re340/re340_36.pdf>.

BOJÓRQUEZ, Gilberto; BOJÓRQUEZ, Jesús. “Metodología para la implementación de sistemas difusos tipo Mamdani en lenguajes de programación de propósito general”. En: *Congreso Internacional en Ingeniería Electrónica. Memoria Electro*. Chihuahua, México: Instituto Tecnológico de Chihuahua, 2014, p. 318-323. ISSN: 1405-2172. Disponible en: <http://depi.itchihuahua.edu.mx/display/memorias_electro/MemoriaElectro2014>.

BRAZ, Marcio R.; VERGILIO, Silvia R.. “Using Fuzzy Theory for Effort Estimation of Object-Oriented Software”. En: *Tools with Artificial Intelligence, 16th IEEE International Conference*. IEEE, 2004, p. 196-201. ISSN: 1082-3409. Florida, Estados Unidos. Disponible en: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1374187>.

BRAZ, Marcio R.; VERGILIO, Silvia R.. “Software effort estimation based on use cases”. En: *30th Annual International, Computer Software and Applications Conference*, *COMPSAC'06*. IEEE, 2006, p. 221-228. ISSN: 0730-3157. Chicago, Illinois, Estados Unidos. Disponible en: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4020081>.

BUSQUELLE, Juan. E. “Análisis de Puntos de Función”. *Lámpsakos*. 2010, no. 4, p. 59-61. Medellín, Colombia. ISSN 2145-4086. Disponible en: < http://www.funlam.edu.co/lampsakos/n4/n4a10.pdf>.

CHEN, Qingzhang; CHENG, Ron; FAN, Shuojin; OU, Yanqiang. "Study of Function Points Analysis Based on Fuzzy-Interpolation”. *Journal of Computational Information Systems*, 2010, vol. 6, no. 5, p. 1369-1375. Hong Kong, República Popular China. ISSN: 1553-9105. Disponible en: <http://www.jofcis.com/publishedpapers/2010_6_5_1369_1375.pdf>.

CMMI Product Team, *CMMI for Development, Version 1.3*. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Estados Unidos. 2010. Disponible en: <http://www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm>.

CONDORI FERNÁNDEZ, Nelly. “Un Procedimiento de Medición de Tamaño Funcional para Especificaciones de Requisitos”. Dirigido por: Oscar Pastor López, Silvia Abrahao. Tesis Doctoral. Universidad Politécnica de Valencia, Departamento de Sistemas Informáticos y Computación, 2007. Valencia, España. Disponible en: <http://www.dsic.upv.es/docs/bib-dig/tesis/etd-01292007-183804/TesisNelly.pdf>.

CONSEJO DE MINISTROS. *Decreto No. 327, "Reglamento del proceso inversionista"*. *Gaceta Oficial de la República de Cuba*, 2015, Número 5, p. 27. La Habana, Cuba. 2015. ISSN: 1682-7511. Disponible en: <http://www.gacetaoficial.cu/pdf/GO_X_5_2015.rar>.

DCG, David Consulting Group. *Informe 2012 Métodos Utilizados en la Industria para Medir el Software*. Laboratorios de las Tecnologías de la Información, 2013. Malvern, Pennsylvania, Estados Unidos. Disponible en: <http://www.laboratorioti.com/2013/02/18/informe-2012-metodos-utilizados-en-la-industria-para-medir-el-software/>.

DE SOUZA LIMA JR, Osias; MUNIZ FARIAS, Pedro P.; DIAZ BELCHIOR, Arnaldo. "Fuzzy modeling for function points analysis". *Software Quality Journal*, 2003, vol. 11, no. 2, p. 149–166. ISSN: 1573-1367. New York, Estados Unidos. Disponible en: <http://link.springer.com/article/10.1023/A:1023716628585>.

DOLADO, Javier; FERNÁNDEZ, Luis. "¿Merece la pena usar los puntos de función?". *Revista Novática de la Asociación de Técnicos de Informática*. 1999, no. 140, p. 57-62. Barcelona, España. ISSN: 0211-2124. Disponible en: < http://www.sc.ehu.es/jiwdocoj/docs/novatica99b.pdf>.

DOMINGO AJENJO, Alberto. *Dirección y Gestión de Proyectos. Un enfoque práctico*. Segunda Edición. México: Alfaomega, 2005, p. 331. Distrito Federal, México. ISBN: 9701511301.

FAN, Wang; XIAOHU, Yang; XIAOCHUN, Zhu; LU, Chen. "Extended Use Case Points Method for Software Cost Estimation". En: *International Conference on Computational Intelligence and Software Engineering*. IEEE, 2009, p. 1–5. Wuhan, China. ISBN: 978-1-4244-4507-3. Disponible en: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5364706>.

FERREIRA LORENZO, Gheisa L; GALVEZ LIO, Daniel; QUINTERO DOMINGUEZ, Luis A.; ANTÓN VARGAS, Jarvin. "Estimación del esfuerzo en proyectos de software utilizando técnicas de inteligencia artificial". *Revista Cubana de las Ciencias Informáticas*, 2014, vol. 8, no. 4, p. 1-20. Ciudad de La Habana, Cuba. ISSN: 2227-1899. Disponible en: <http://rcci.uci.cu/index.php?journal=rcci&page=article&op=view&path[]=717>.

HENDERSON, David; SHEETZ, Steven D.; WALLACE Linda. "Understanding Software Metric Use". *ISACA Journal online*, 2015, volume 1. Rolling Meadows, Illinois, Estados Unidos. Disponible en: <http://www.isaca.org/Journal/archives/2015/Volume-1/Pages/Understanding-Software-Metric-Use.aspx>.

HERNÁNDEZ SAMPIERI, Roberto; FERNÁNDEZ-COLLADO, Carlos; BAPTISTA LUCIO, Pilar. *Metodología de la Investigación*. Sexta edición. Distrito Federal, México: McGraw Hill, 2014, p. 600. ISBN 9781456223960.

IFPUG, THE INTERNATIONAL FUNCTION POINT USERS GROUP. *Function Point Counting Practices Manual*. Release 4.3.1. 2010. New Jersey, Estados Unidos. ISBN 978-0-9753783-4-2. Disponible en: <http://ainfo.cnptia.embrapa.br/digital/bitstream/item/34989/1/0004-3-1-Part-0-2010-01-17.pdf>

JASSBI, Javad; ALAVI, Sajid H.; RIBEIRO, Rita A.. “Transformation of a Mamdani FIS to first order sugeno FIS”. En: *IEEE international conference on Fuzzy systems*. London, United Kingdom: IEEE International, 2007, p. 1-6. ISBN: 1-4244-1209-9. Disponible en: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4295331>.

LA RED MARTÍNEZ, David L.; CÉSAR ACOSTA, Julio. “Revisión de Operadores de Agregación”. *Revista Campus Virtuales*, 2014, Vol. 3, no. 2, p. 24-44. Huelva, España. ISSN: 2255-1514. Disponible en: <http://www.uajournals.com/campusvirtuales/journal/5/3.pdf>.

LEE, Chuen-Chien. “Fuzzy Logic in Control Systems: Fuzzy Logic Controller-Part II”. *Systems, Man and Cybernetics*. IEEE Transactions, 1990, vol. 20, no. 2, p. 419-435. New York, Estados Unidos. ISSN: 0018-9472. Disponible en: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=52552>.

LEE, Jonathan; LEE, Wen-Tin; KUO, Jong-Yin. “Fuzzy logic as a basic for use case point estimation”. En: *IEEE International Conference on Fuzzy Systems*. IEEE, 2011, p. 2702-2707. Taipei, Taiwan. ISSN: 1098-7584. Disponible en: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6007717>.

MACHADO PÍRIZ, Fernando. “Conversión COSMIC/IFPUG”. Dirigido por: Juan José Cuadrado Gallego. Tesis Doctoral. Universidad de Alcalá de Henares, Departamento de Ciencias de la Computación, 2006. Madrid, España. Disponible en: <http://dialnet.unirioja.es/servlet/tesis?codigo=1602>.

MAGDALENA, Luis. “What is Soft Computing? Revisiting Possible Answers”. *International Journal of Computational Intelligence Systems*, 2010, vol. 3, no. 2, p. 148-159. París, Francia. ISSN: 1875-6883. Disponible en: <http://www.tandfonline.com/doi/abs/10.1080/18756891.2010.9727686>

MAGUIÑA, Rolando. A.. “Sistemas de inferencia basados en Lógica Borrosa: Fundamentos y caso de estudio”. *Revista de Investigación de Sistemas de Informática*, 2010, vol. 7, no. 1, p. 91-104. Lima, Perú. ISSN: 1816-3823. Disponible en: <http://revistasinvestigacion.unmsm.edu.pe/index.php/sistem/article/view/3270>.

MANIFESTO, Chaos. *Think Big, Act Small*. The Standish Group International Inc, 2013. West Yarmouth, Massachusetts, Estados Unidos. Disponible en: <http://versionone.com/assets/img/files/ChaosManifesto2013.pdf>.

MARTÍNEZ LÓPEZ, Luis. “Un nuevo modelo de representación de información lingüística basado en 2-tuplas para la agregación de preferencias lingüísticas”. Dirigido por: Francisco Herrera Triguero. Tesis Doctoral. Universidad de Granada, Departamento de Ciencias de la Computación e Inteligencia Artificial, 1999. Granada, España. Disponible en: <http://sinbad2.ujaen.es/cod/archivosPublicos/tesis/pdf/TesisLuisMartinez.pdf>.

NASSIF, Ali B; Capretz, Luiz F.; HO, Danny. “Enhancing Use Case Points Estimation Method using Soft Computing Techniques”. *Journal of Global Research in Computer Science*, 2010, vol. 1, no. 4, p. 12–21. Illinois, Estados Unidos. ISSN: 2229-371X. Disponible en: <http://jgrcs.info/index.php/jgrcs/article/view/174>.

NASSIF, Ali B.; HO, Danny ; Capretz, Luiz F.. “Regression model for software effort estimation based on the use case point method”. En: *International Conference on Computer and Software Modeling*. 2011 a, p. 117–121. New Delhi, India. ISBN : 978-981-08-9917-2. Disponible en: <http://www.academia.edu/download/30512112/20-iccsm2011-s0054.pdf>

NASSIF, Ali B.; Capretz, Luiz F.; HO, Danny. "Estimating Software Effort Based on Use Case Point Model Using Sugeno Fuzzy Inference System". En: *IEEE 23rd International Conference on Tools with Artificial Intelligence*. IEEE, 2011 b, p. 393-398. Florida, Estados Unidos. ISBN 978-1-4577-2068-0. Disponible en: <http://doi.ieeecomputersociety.org/10.1109/ICTAI.2011.64>.

NASSIF, Ali B.; Capretz, Luiz F.; HO, Danny. “Calibrating use case points”. En: *Proceedings of the 36th International Conference on Software Engineering*, 2014, p. 612–613. Hyderabad, India. ISBN: 978-1-4503-2768-8. Disponible en: <http://www.researchgate.net/profile/Ali_Nassif2/publication/263084056_Calibrating_Use_Case_Points/links/00463539bc31ca0bab000000.pdf>.

NEIL THOMPSON, Donaval. “Proyectos Informáticos: Fracasos y Lecciones Aprendidas”. *Revista de Derecho y Tecnologías de la Información*, 2006, no. 4. Universidad Estatal a Distancia, Atenas, Costa Rica. Disponible en: <http://tg-tatiana-oquendo.googlecode.com/svn/trunk/art8.pdf>.

NUNES, Nuno J.; CONSTANTINE, Larry; KAZMAN, Rick. *IUCP: Estimating interactive-software project size with enhanced use-case points*. *IEEE Software*, 2011, vol. 28, no. 4, p. 64-73. California, Estados Unidos. ISSN: 0740-7459. Disponible en: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5499458>.

PCC, Partido Comunista de Cuba. “Lineamientos de la Política Económica y Social del Partido y la Revolución”. En: *VI Congreso del Partido Comunista de Cuba*. La Habana, Cuba. 2011. Disponible en: <http://www.juventudrebelde.cu/file/pdf/suplementos/lineamientos-politica-partido-cuba.pdf>.

PEREGRÍN RUBIO, Antonio. “Integración de operadores de implicación y métodos de defuzzificación en sistemas basados en reglas difusas. Implementación, análisis y caracterización”. Dirigido por: Francisco Herrera Triguero. Tesis Doctoral, Universidad de Granada, Departamento de Ciencias de la Computación e Inteligencia Artificial, 2000. Granada, España. Disponible en: <http://decsai.ugr.es/Documentos/tesis_dpto/51.pdf>.

PIÑERO PÉREZ, Pedro. “Un modelo para el aprendizaje y la clasificación automática basado en técnicas de soft computing”. Dirigido por: María Matilde García Lorenzo. Tesis Doctoral. Universidad Central Marta Abreu de las Villas, Departamento de Ciencias de la Computación, 2005. Villa Clara, Cuba.

REMÓN, Cristian A.; THOMAS, Pablo J.. “Análisis comparativo de Estimación de Esfuerzo en el Desarrollo de Software”. En: *XVII Congreso Argentino de Ciencias de la Computación*, VIII Workshop Ingeniería de Software (WIS), 2011, p. 729-738. La Plata, Argentina. ISBN: 978-950-34-0756-1. Disponible en: <http://sedici.unlp.edu.ar/bitstream/handle/10915/18729/Documento_completo.pdf?sequence=1>.

SALAZAR-B, Gabriela. “Estimación de proyectos de software: un caso práctico”. *Ingeniería y Ciencia*. 2009, vol. 5, no. 9, p. 123-143. Medellín, Colombia. ISSN 1794-9165. Disponible en: <http://publicaciones.eafit.edu.co/index.php/ingciencia/article/view/470/437>.

SCHNEIDER, Geri; WINTERS, Jason P.. *Applying Use Cases: A practical Guide*. Segunda Edición. Estados Unidos: Addison Wesley, 2001, p. 272. The Addison-Wesley Object Technology Series. ISBN: 0-201-70853-1.

XIA, Wei; HO, Danny; CAPRETZ, Luis F.. “A Neuro-Fuzzy Model for Function Point Calibration”. *WSEAS Transactions on Information Science & Applications*, 2008, Issue 1, Volume 5. Wisconsin, Estados Unidos. ISSN: 1790-0832. Disponible en: <http://www.researchgate.net/profile/Danny_Ho/publication/234815348_A_neuro-fuzzy_model_for_function_point_calibration/links/09e415112d52e6e166000000.pdf>

YAU, Chuk; TSOI, Ho-Leung. “Modelling the probabilistic behaviour of function point analysis”. *Information and Software Technology*, 1998, vol. 40, no. 2, p. 59–68. Karlskrona, Sweden. ISSN: 0950-5849. Disponible en: <http://www.sciencedirect.com/science/article/pii/S0950584998000329>.

ZADEH, Lotfi A.. What is *Computing with Words (CWW)?*. Springer Berlin Heidelberg, 2012. Studies in Fuzziness and Soft Computing, vol. 277, p. 142. Berlín, Alemania. ISBN: 978-3-642-27472-5.

ZULUETA VELIZ, Yeleny. “Modelos de evaluación de la importancia del impacto ambiental en contextos complejos bajo incertidumbre”. Dirigida por: Luis Martínez López. Tesis Doctoral. Universidad de Granada, Departamento de Ciencias de la Computación e Inteligencia Artificial, 2014. Granada, España. Disponible en: <http://sinbad2.ujaen.es/cod/archivosPublicos/tesis/pdf/TesisYeleny.pdf>