novaRua 90

Modelo de evaluación de proyectos

de *software* utilizando una lógica difusa compensatoria

Héctor Medina Aguilar
Liliana Angélica Guerrero Ramos
Rafael Alejandro Espín Andrade

Universidad Autónoma
de Coahuila

Recibido: 8 de mayo de 2017
Aceptado: 28 de agosto de 2017

Resumen

Hoy en día el *software*, en cualquier empresa, es una herramienta primordial para el desarrollo de casi cualquier actividad. Sin embargo, contrario a lo que se piensa acerca de la gestión de proyectos de *software*, este es un proceso complicado en el cual se presentan diversas variables que inciden en las diferentes etapas de un proyecto: su gestión en general, el análisis, el desarrollo y la implementación. Esto provoca que un gran porcentaje de proyectos no logren alcanzar la finalización o cumplir con las funcionalidades básicas necesarias, ocasionando problemas para las corporaciones. El objetivo de esta investigación es generar un modelo general para la evaluación de proyectos de *software* utilizando una lógica difusa compensatoria, con el objetivo de tener un análisis de los factores que influyeron en los resultados de los proyectos de *software*. La metodología empleada es exploratoria y explicativa, ya que se analizará la documentación y se tomarán como base investigaciones previas en las que se utilice una

lógica difusa compensatoria como herramienta de análisis. Como resultado, se obtuvo y aplicó el modelo difuso para la evaluación de tres proyectos de *software*, a modo de prueba piloto.

Palabras clave: evaluación de proyectos de *software*, lógica difusa compensatoria.

Abstract

Nowadays software in any organization is a prime tool for the development of almost any business activity. However contrary to what is thought about the management of software projects, this is a complicated process in which several variables that affect the different stages of a project are presented: its management in general, analysis, development, and implementation. This causes that a large percentage of projects fail to achieve completion or fulfill the necessary basic functionalities causing problems for corporations. The objective of this research is to generate a general model for evaluation of software projects using a compensatory diffuse logic in order to have an analysis of the factors that influenced the results of software projects. The methodology used is exploratory and explanatory because documentation will be analyzed and based on previous research using a compensatory diffuse logic as an analysis tool. As a result, the diffuse model was obtained and applied to evaluate three software projects as a pilot test.

Keywords: software projects evaluation, compensatory diffuse logic.

Introducción

> "El *Software* no se elabora como cualquier producto industrial, no se construye, se desarrolla e involucra intensamente a la gente al ser una actividad mental y creativa" (Zavala, 2014).

Actualmente, los sistemas de información son un elemento de suma importancia en casi todos los aspectos de la vida cotidiana, debido a que los podemos encontrar en la industria, la salud, la educación, el entretenimiento, el deporte, etcétera. Casi cualquier aspecto de nuestra vida tiene relación con información que se encuentra registrada, procesada o transferida entre sistemas de información; he ahí la importancia de estos.

Centrándonos en la industria del *software*, esta tiene un peso importante en cualquier área empresarial, como lo mencionamos anteriormente, ya que no solo es una unidad de soporte para las actividades principales de las compañías, sino que es un área en donde se generan tanto innovación como mejoras para los diferentes procesos; es ahí donde recae la importancia de tener *software* para mejorar la calidad y en los tiempos en los que las organizaciones lo requieran.

Uno de los grandes problemas de la industria del *software*, es que, a pesar de que hay estándares, metodologías, técnicas, lineamientos y demás herramientas, estas no se emplean de manera generalizada, haciendo de aquella algo menos que una artesanía. Además, los profesionales en *software*, en su gran mayoría, tienen deficiencias académicas importantes y muchos de ellos son generalistas —o todólogos— en vez de especialistas. Con los esquemas *ad hoc* que adopta la industria, no se puede repetir ni predecir el proceso de producción ni estimar la calidad del producto final (Zavala, 2014).

En alineación con esta situación, el problema de la investigación es la carencia o falta de uso extensivo de metodologías para evaluar los proyectos de desarrollo de *software*.

La lógica difusa (LD), que es una técnica de la inteligencia computacional que permite modelar situaciones vagas en los procesos de toma de decisiones, ha sido empleada en diversos campos; particularmente, en el tema de la gestión de proyectos de *software*, se ha usado para estimar el esfuerzo en su implementación (Ferreira, Gálvez, Quintero, & Antón, 2014), así como para la evaluación de riesgos en los mismos (Rodríguez, Ortega, & Concepción, 2016), pero no se identificó un modelo difuso para su evaluación integral.

El objetivo de la investigación es, por lo tanto, proponer un modelo general para la evaluación de proyectos de *software* utilizando una lógica difusa compensatoria (LDC), con el propósito de tener un análisis de los factores que influyeron en los resultados de los proyectos de *software* y estar en mejores condiciones de poder tomar decisiones correctivas.

Los hallazgos más importantes de esta investigación radican en la realidad identificada, al argumentar la importancia de la misma, que compara otros métodos tradicionales de toma de decisiones con la LD, observándose las ventajas que esta última tiene para representar problemas del lenguaje natural o profesional en cualquier ámbito de análisis y la propuesta en sí misma del modelo difuso para la evaluación de proyectos de *software*.

La investigación está estructurada, iniciando con el análisis de estudios realizados y fundamentos teóricos relacionados con el tema a investigar; la segunda parte es la metodología utilizada, en donde se identifican los criterios a evaluar y se explica el uso del *software* Fuzzy Tree Studio para el diseño del modelo, a efectos de cumplir con el objetivo propuesto; en la tercera parte, se hace una descripción de los resultados obtenidos con el diseño del modelo y la evaluación de tres proyectos; y, finalmente, se presenta una discusión y trabajo futuro, además de conclusiones y referencias bibliográficas.

Marco teórico

¿Qué es la ingeniería de *software*?

Es una disciplina que comprende todos los aspectos de la producción de *software*, desde las etapas iniciales de la especificación del sistema hasta el mantenimiento de este después de que se utiliza.

En general, los ingenieros de *software* adoptan un enfoque sistemático y organizado en su trabajo, ya que es la forma más efectiva de producir *software* de alta calidad; sin embargo, aunque la ingeniería de *software* consiste en seleccionar el método más apropiado para un conjunto de circunstancias, un enfoque más informal y creativo de desarrollo podría ser efectivo en algunas situaciones (Sommerville, 2000).

¿Qué es un modelo de procesos de *software*?

Es una descripción simplificada de un proceso de *software*, que presenta una visión del mismo. Estos modelos pueden incluir actividades que son parte de

los procesos y productos de *software*, además del papel de las personas involucradas en la ingeniería de *software*:

- Modelo de flujo de trabajo: muestra la secuencia de actividades en el proceso de juntar entradas, salidas y dependencias. Las actividades en este modelo representan acciones humanas.
- Modelo de flujo de datos o actividades: representa el proceso como un conjunto de actividades, donde cada una realiza alguna transformación en los datos.
- Modelo de rol/acción: representa los roles de las personas involucradas en el proceso de *software* y las actividades de las que son responsables.

La mayor parte de los modelos de procesos de *software*, se basan en uno de los tres modelos generales o paradigmas de desarrollo de *software*:

1) Enfoque en cascada: considera las actividades anteriores y las representa como fases de procesos separados, tales como la especificación de requerimientos, el diseño de *software*, la implementación y las pruebas.
2) Desarrollo iterativo: este enfoque entrelaza las actividades de especificación, desarrollo y validación. Un sistema inicial se desarrolla rápidamente a partir de especificaciones muy abstractas. Este se refina basándose en las peticiones del cliente para producir un sistema que satisfaga las necesidades de aquel.
3) Ingeniería de *software* basada en componentes (CBSE): supone que las partes del sistema existen; el proceso de desarrollo del sistema, se enfoca en la integración de esas partes más que en desarrollarlas desde el principio.

¿Qué es el *software*?

Según Zavala (2014), en su artículo ¿Por qué fracasan los proyectos de *software*?; un enfoque organizacional, estas son las razones:

a) El *software* es el activo más importante de las corporaciones, aunque pocas veces se le otorgue un valor distinto al de su uso;
b) El *software* se ha convertido en el corazón de la operación de la empresa y prácticamente no hay algo que no esté relacionado con su operación, aunque en muchas compañías las actividades informáticas son consideradas como una actividad "de apoyo";
c) El *software* es tan importante que una falla de este, puede paralizar a la organización entera y a sus socios de negocios. Todo crecimiento de la empresa implica un crecimiento del *software* y de los requerimientos para su desarrollo. La factibilidad de implementación de las políticas de negocios, pasa a depender de la capacidad del *software* y del personal que lo opera para gestionar los datos acordes a la misma.

Proyectos de *software*

La mayor parte se desarrollan por equipos de desarrollo del proyecto, desde unas cuantas personas hasta grandes grupos de varias decenas de individuos. Es personal altamente especializado y calificado con actividades prácticamente intelectuales y creativas.

La percepción común que se tiene de la industria del *software*, es que es sólida, pero

sorprendentemente no se caracteriza por la alta calidad generalizada de sus productos y servicios. La investigación que más se cita sobre el estado de los proyectos de *software*, es el famoso "Reporte Caos" del Standish Group, que prácticamente todos los investigadores asumen como referencia obligada. El Standish Group (2014) clasifica los proyectos en tres tipos:

- Exitoso (*successful*: el proyecto se completa en tiempo y dentro del presupuesto, con todas las características y funciones
- Desafiante (*challenged*): el proyecto se completa y es operacional, pero más allá del presupuesto y del tiempo estimado, y con pocas de las características y funciones que fueron especificadas inicialmente
- Fracasado (*failed*): el proyecto es cancelado antes de completarse

Es indudable que los proyectos de *software*, se caracterizan por altas tasas de fracaso o falla. Y, a pesar de ello, el mito de que la industria es de una alta tecnología persiste.

Gestión de proyectos de *software*

"La gestión de proyectos de software persigue la misma finalidad que todas las gestiones de proyectos de ingeniería, estimar que sucederá con un proyecto nuevo y analizar que sucedió con un proyecto ya finalizado" (Grompone, 1996).

La gestión de proyectos de *software*, es una rama especializada de la ingeniería de *software*.

Según Pressman (2002), la gestión eficaz de un proyecto de *software*, se centra en cuatro P:

a) Personal;
b) Producto;
c) Proceso; y
d) Proyecto.

Síntomas de una entrega deficiente

Para McManus y Wood-Harper (2003), los síntomas de una entrega deficiente de los sistemas de información son los siguientes:

- Solicitudes de cambio frecuentes por usuarios
- Los usuarios tienen un deficiente entendimiento de sus propias necesidades
- Tareas no consideradas
- Comunicación insuficiente
- Deficiencia en una metodología adecuada y lineamientos para su estimación
- Deficiencia de coordinación del desarrollo de sistemas
- Tiempo insuficiente para pruebas
- Deficiencia en la preparación
- Alineación de la estrategia de negocios deficiente

Factores de éxito en los proyectos

Los factores que afectan el éxito de los proyectos, según Baker, Murphy y Fisher, citados por McManus y Wood-Harper (2003), quienes estudiaron seiscientos cincuenta proyectos en Estados Unidos, son los siguientes:

- Compromiso con el proyecto en el establecimiento de calendarizaciones, presupuestos y objetivos de desempeño técnicos
- Frecuente retroalimentación de la empresa patrocinadora
- Frecuente retroalimentación del cliente

- Compromiso del cliente y del patrocinador comprometidos en el establecimiento de calendarizaciones, presupuestos y objetivos de desempeño técnicos
- Estructura de la organización adecuada al equipo del proyecto
- Participación del equipo del proyecto en la determinación de las calendarizaciones y los presupuestos
- Entusiasmo del patrocinador
- Deseo del patrocinador de crear capacidades internas
- Procedimiento de controles adecuados, especialmente en relación con los cambios
- Uso con juicio de técnicas de programación en red
- Mínimo de agencias públicas y de gobierno involucradas
- Falta de un gobierno excesivo
- Soporte de un público entusiasta
- Falta de impedimentos legales

Factores críticos para el éxito de un proyecto propuestos por Pinto (1990):

Misión del proyecto: metas y direcciones generales definidas con claridad al inicio del proyecto

- Soporte administrativo de alto nivel: ayuda de la alta dirección para proveer los recursos necesarios, además de autoridad y poder para el éxito del proyecto
- Auscultación del cliente: comunicación, auscultación y escucha activa de todas las partes impactadas
- Personal: reclutamiento, selección y entrenamiento del personal necesario para el equipo del proyecto
- Tareas técnicas: disponibilidad de la tecnología y experiencia necesarias para el cumplimiento de las acciones técnicas específicas
- Aceptación del cliente: acto de "vender" el final del proyecto a los usuarios finales
- Monitoreo y retroalimentación: provisión a tiempo y de manera adecuada de información de control en cada una de las etapas del proceso de implementación del proyecto
- Comunicación: provisión de una red apropiada y de datos necesarios para todos los actores clave en la implementación del proyecto
- Resolución de problemas: habilidad para manejar crisis inesperadas y desviaciones del plan

Modelos de decisión para la evaluación de proyectos de *software* con énfasis en el uso de la inteligencia artificial

En la evaluación de proyectos de *software* utilizando técnicas de inteligencia artificial, ha recibido especial atención en los últimos años la estimación del esfuerzo. Una investigación que revisa los modelos más utilizados, plantea que estos son: Razonamiento basado en casos (RBC); Algoritmos genéticos (AG); Programación genética (PG); Regresión del vector de soporte (en inglés, support vector regression – SVR); Redes neuronales artificiales (RNA); Sistemas de inferencia borrosos (SIB) y Lógica difusa (LD); y Árboles de decisión (AD) (Ferreira *et al.*, 2014). Lo cual impresiona al ser un tema de gran relevancia; sin embargo, otros trabajos recientes de

investigación desarrollan metodologías basadas en variadas técnicas de inteligencia computacional (Remón, & Thomas, 2010; López, 2017; Velarde, & Santiesteban, 2017).

> La estimación temprana del esfuerzo para la construcción de un producto de software, es crucial en la previsión del costo y tiempo necesarios para su desarrollo. Los modelos y técnicas para la estimación del esfuerzo presentan como principal inconveniente. La poca precisión de las predicciones realizadas y generalmente se hace una mínima consideración de los aspectos no funcionales del software (Almache, Raura, Ruiz, & Fonseca, 2015, p. 148).

Estos autores desarrollaron esta aplicación entrenando una red neuronal de aprendizaje con datos de aplicaciones del ámbito académico, en donde se conocían el tiempo y los costos incurridos en los proyectos.

Una reciente investigación realiza una propuesta para determinar los factores críticos en el éxito de los proyectos de *software* y así asignarles una ponderación, para lo cual se emplean técnicas de grupo focal y un proceso de jerarquía analítica (en inglés, analitic hierarchy process [AHP]). En los resultados, se mostró la importancia que recibe el compromiso del cliente con respecto a otros factores (Peña, Barrionuevo, & Cedeño, 2016).

Lógica difusa compensatoria y su importancia en aplicaciones para la evaluación de proyectos de *software*

La LD fue formulada por el matemático e ingeniero Lotfi A. Zadeh, profesor de la Universidad de California en Berkeley. Es una disciplina que surgió motivada por el estudio de la vaguedad, de la información vaga o de difícil especificación, aunque también permite estudiar y modelar procesos de toma de decisiones con un alto nivel de incertidumbre; pero vaguedad e incertidumbre son conceptos diferentes, ya que la incertidumbre está asociada al desconocimiento del valor de una función de una variable, mientras que la vaguedad está relacionada con el conocimiento del valor de una función (llamada grado de pertenencia) de una variable, cuyo valor exacto se conoce. En otras palabras, la LD es una técnica de la inteligencia computacional que permite trabajar con un alto grado de imprecisión y que trata de copiar la forma en la que los humanos toman decisiones (D'Negri, & De Vito, 2006).

La LDC es una aproximación lógica multivariada diferente a la norma y con forma axiomática. Es una teoría lógica transdisciplinaria enfocada en un propósito definido como la interpretabilidad de acuerdo al lenguaje, lo que le confiere una gran fortaleza como herramienta óptima para generar modelos administrativos empleando la ingeniería del conocimiento. La interpretabilidad ocurre de acuerdo con teorías lógicas y paradigmas asociados con muchas prácticas sociales en relación con el lenguaje natural y profesional, así como con la lógica clásica, teorías y métodos de toma de decisiones, estadística matemática y otras disciplinas y campos de cono-

cimiento (Espín, González, Pedrycz, & Fernández, 2016).

La LDC arquimediana es compatible con el enfoque clásico de la norma, según se expone en un artículo que introduce este nuevo aporte a la LD (Espín, González, Pedrycz, & Fernández, 2015).

Esta aplicación en conjunto con otras desarrolladas o que están en desarrollo, utilizando un *software* denominado Fuzzy Tree Studio, que será explicado posteriormente, u otras opciones de *software* como instrumentos analíticos, están permitiendo integrar una metodología de inteligencia organizacional semántica para la gestión del conocimiento y toma de decisiones.

El *software* Fuzzy Tree Studio tiene su antecedente en el iC Pro, presentado por primera vez en 2008 por profesores de la Universidad Nacional de Mar del Plata, Argentina, dirigidos por el ingeniero Gustavo Meschino, conceptualizado como un *framework* de análisis de datos con técnicas de inteligencia computacional. Este *software* facilitó el cálculo de los valores de verdad asociados con modelos basados en LDC. Como un desarrollo posterior para superar limitaciones del iC Pro, surgió por los mismos creadores el *software* Fuzzy Tree Studio, "que entre otras funcionalidades posee un módulo para ayudar al usuario a formalizar y calcular el valor de verdad de predicados parciales y operar adecuadamente con ellos, generalizando los conceptos de la Lógica de predicados tradicional" (Chao-Ballester, & Espín-Andrade, 2015, p. 171).

En la búsqueda de aplicaciones desarrolladas con base en la LDC, no se encontraron trabajos que desarrollen una metodología completa como la que se propone en esta investigación; sin embargo, sí existen algunas aplicaciones basadas en LD para evaluar ciertos parámetros. Tal es el caso de una investigación que evalúa la usabilidad, que se define como el grado de eficacia, eficiencia y satisfacción con el que ciertos usuarios pueden lograr objetivos de uso específico, la cual es analizada en términos de comprensibilidad, aprendizaje, operabilidad, atractivo y complacencia.

La usabilidad tiene atributos cuantificables de manera objetiva, por ejemplo, el número de errores cometidos por el usuario en la realización de una tarea, así como atributos cuantificables de manera subjetiva, como la satisfacción de uso en estrecha relación con la usabilidad percibida (Baquero, Rodríguez, & Ciudad, 2016).

La evaluación de riesgos en proyectos de tecnologías de la información, también ha sido objeto de investigaciones en las que se plantea que estos proyectos son particularmente propensos al fracaso, debido a sus características específicas, haciendo que la evaluación de riesgos se convierta en un elemento crítico en su gestión, por lo que se propone un nuevo método de evaluación de riesgos basado en una combinación del proceso de jerarquía analítica difusa (FAHP, por sus siglas en inglés) y el sistema de inferencia difusa (FIS, por sus siglas en inglés) (Rodríguez, Ortega, & Concepción, 2016).

Por otra parte, en el propio desarrollo de ciertos proyectos de procesos productivos existen varios ejemplos de aplicaciones de la LD. Tal es el caso del manejo de un algoritmo genético para minimizar el

tiempo del proceso de corte en una empresa que confecciona dotaciones industriales, dentro del cual se implementó y desarrolló el concepto de LD, mediante el cual variables borrosas, tales como la habilidad de los operarios, el patrón de corte y las características del material, se volvieron variables reales (Duarte, & Orozco Ahumada, s.f.).

Con el propósito de argumentar la relevancia de las aportaciones de esta investigación, se consultó la obra *Las claves de la argumentación*, de Anthony Weston (2006), en donde se expone la argumentación como la exposición de un conjunto de razones en apoyo de una conclusión, además de que se ofrecen algunas recomendaciones, entre otras, preguntarse primero: ¿qué se está tratando de probar? y ¿cuál es la conclusión? Presentar las ideas en un orden natural, partir de premisas fiables, ser concreto y preciso, evitar un lenguaje emotivo y utilizar un significado único para cada término.

Lo que se está tratando de probar en este trabajo, es que la LDC y el programa Fuzzy Tree Studio ofrecen una plataforma de gran utilidad para diseñar un modelo de evaluación de proyectos de *software* y la conclusión fundamental es que la LDC, en su aplicación mediante el uso del programa Fuzzy Tree Studio, provee instrumentos analíticos que permiten modelar cualquier problema del lenguaje natural o profesional.

Una premisa fiable, concreta y precisa en esta dirección, se ofrece al hacer una comparación entre modelos clásicos de toma de decisiones multicriterio y de LD, cuando se plantea que aunque el AHP ha sido aplicado a diferentes situaciones con resultados razonables, no es capaz de abordar la complejidad inherente a muchos problemas del mundo real, debido a su estructura jerárquica; sin embargo, otros modelos tratan de mejorar este problema. No obstante, cuando se trata de integrar variables lingüísticas, que son las que abundan en la mayoría de los problemas de la vida real, la LD es la técnica apropiada (Rodríguez, 2008).

Metodología

Se trata de una investigación con enfoque cualitativo, debido a que se explicaron conceptos referentes a proyectos de *software*, además de explorar en la documentación diferentes estudios que se han realizado para definir las razones por las que este tipo de proyectos tienen éxito o fracasan. Se recopilaron diferentes estudios y material estadístico referentes a proyectos de *software*, con la finalidad de generar un modelo de evaluación de estos proyectos. Además, se analizaron diferentes artículos en los cuales se utiliza la LD como medio de análisis de resultados.

Se diseñaron los diagramas de evaluación que se muestran en la primera sección de los resultados con el *software* Fuzzy Tree Studio. Los fundamentos de la LDC y su interpretabilidad, se pueden consultar en Espín *et al.* (2015) y Espín *et al.* (2016). La metodología para gestionar el conocimiento basada en LDC, así como ejemplos del uso del *software* Fuzzy Tree Studio, se pueden consultar en Chao-Ballester, & Espín-Andrade (2015).

Fuzzy Tree Studio es un sistema de soporte de decisiones basado en árboles con operadores de LD desarrollado en la Universidad Nacional de Mar del Plata, Argentina (Ge-

sualdo, 2010), la cual posee un módulo que trabaja con LDC. Este sistema facilita no tener que emplearse en el trasfondo matemático y poder centrarse en la formulación verbal en lenguaje natural o profesional, así como en la construcción de un modelo que permita tomar decisiones (Cejas, 2011; Meschino, Nabte, Gesualdo, Monjeau, & Passoni, 2014).

Se aplicó la evaluación de diferentes proyectos de *software* utilizando los siguientes criterios de clasificación:

- Tipo de proyecto:

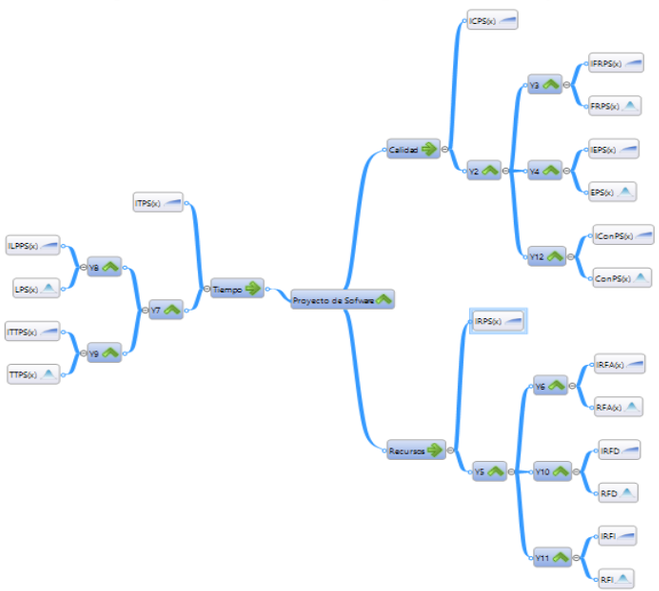
a) Mejora (*enhancement request*);
b) Solución del problema (*break fix*); o
c) Proyecto nuevo (*new project*).

- Tamaño del proyecto:

a) Grande (*big project*);
b) Mediano (*medium project*); o
c) Pequeño (*small project*).

- Prioridad:
- Alta (*high*);
- Media (*medium*); o
- Baja (*slow*).

Gráfico 1. Diagrama de evaluación de proyectos de *software*.



Fuente: elaboración propia.

Con base en las clasificaciones anteriores, se analizaron diferentes proyectos sin importar el objetivo o impacto organizacional, sino la eficiencia en la gestión de los mismos.

La descripción del diagrama, se muestra a continuación:

{ { Impacto Calidad de un proyecto de software (x) **entonces** { { Impacto de las funcionalidades requeridas de un proyecto de software (X) **y** Funcionalidades Requeridas de un proyecto de software(x) } , { Impacto de Eficiencia de un proyecto de software(x) **y** Eficiencia de un proyecto de sofware(x) } **y** { Impacto de la confiabilidad de un proyecto de software(x) **y** Confiabilidad de un proyecto de software(x) } } } , { Impacto de los recuersos de un proyecto de software(x) **entonces** { { Impacto recursos asignados a la fase de analisis del proyecto(x) en base al presupuesto **y** Recursos Asignados a la fase de Analisis } , { Impacto recursos asinados a la fase de desarrollo del proyecto (x) en base al presupuesto **y** Recursos asignados en la fase de desarrollo } **y** { Impacto de recusos asignados a la fase de implementacion de un proyecto (x) en base al presupuesto **y** Recuerso Asignados a la pase de Implementacion } } } **y** { Impacto del tiempo de un proyecto de software(x) **entonces** { { Impacto Liberacion parciales del proyecto de software(x) **y** Liberación del proyecto de software(x) } **y** { Impacto del tiempo de termino del proyecto de software(x) **y** Tiempo de termino del proyecto de software } } } }

Tabla 1. Resultados de evaluación de tres proyectos de *software*.

Orden	ICPS(x)	IFRPS(x)	FRPS(x)	IEPS(x)	EPS(x)	IConPS(x)	ConPS(x)	IRPS(x)	IRFA(x)	RFA(x)	IRFD	RFD	IRFI	RFI	ITTPS(x)	ILPPS(x)	LPS(x)	ITTPS(x)	TTPS(x)	GMBCL (Geometric Mean Based Compensatory Logic)
1	.95	.90	80	.80	.75	.90	80	.90	.95	90	.90	80	.35	95	.95	.85	75	.35	80	0.2809
2	.95	.90	90	.80	90	.90	95	.90	.95	90	.90	90	.35	95	.95	.85	90	.35	95	0.4581
3	.95	.90	85	.80	85	.90	85	.90	.95	70	.90	95	.35	95	.95	.85	85	.35	90	0.3562
Operadores cuantificadores																				
																				Existe = 0.3693
																				Para todo = 0.3579

Fuente: elaboración propia.

Análisis e interpretación de resultados

En el gráfico 1, se muestra el diagrama de evaluación de proyectos de *software* elaborado con el programa Fuzzy Tree Studio, que contempla las dimensiones de calidad, recursos y tiempo.

Se evaluaron, a modo de ejemplo, tres proyectos de *software* ya concluidos, cuyos resultados de evaluación se muestran en la tabla 1.

Discusión y trabajo futuro

Las tres macrovariables del modelo difuso de evaluación de proyectos de *software* propuesto (calidad, recursos y tiempo) coinciden con las propuestas de varios autores consultados en la literatura científica sobre el tema (Mendoza, Pérez, & Grimán, 2005; Figueroa, Solís, & Cabrera, 2008; Grompone, 1996; Letelier, & Panadés, 2006; Markus, 2010; Zavala, 2014).

En la evaluación de los tres proyectos de desarrollo de *software*, a modo de prueba piloto, el segundo resultó el mejor evaluado. Como es característico de los modelos de decisiones, se ofrece una jerarquía, pero esta vez basada en los criterios de verdad asignados a cada variable del modelo por los evaluadores.

Se comprobó la funcionalidad que ofrece la LD para representar problemas del lenguaje natural y profesional, logrando un proceso de evaluación y toma de decisiones más comprensible para el usuario, tal y como se plantea en la literatura consultada (Chao-Ballester, & Espín-Andrade, 2015; D'Negri, & De Vito, 2006; Espín *et al.*, 2016; Zapata, & Arango, 2005).

Como trabajo futuro, se precisa continuar con la aplicación del modelo propuesto, afinando su estructura mediante criterios de expertos y diferentes evaluaciones en otros contextos de aplicación. Particularmente, hay una línea importante para continuar, derivada de la observación en la revisión de la literatura, debido a que se encontraron varios trabajos enfocados en la estimación del esfuerzo en los proyectos de *software* (Almache *et al.*, 2015; Ferreira *et al.*, 2014; Remón, & Thomas, 2010; López, 2017; Velarde, & Santiesteban, 2017).

Conclusiones

- A pesar de que en la industria del *software* existen estándares, metodologías,

técnicas, lineamientos y demás herramientas, estas no se emplean de manera generalizada, además de que no existe un consenso de cómo estimar la calidad del producto final

- Algunas de las cuestiones que afectan la calidad en el desarrollo de *software* son: solicitudes de cambio frecuentes por usuarios; poco entendimiento por parte de los usuarios de sus propias necesidades; tareas no consideradas; comunicación insuficiente; deficiencia en metodología adecuada y lineamientos para estimación; y alineación de estrategia de negocios deficiente, entre otras
- Entre los factores que propician el éxito en los proyectos, se encuentran los siguientes: compromiso con el proyecto en el establecimiento de calendarizaciones, presupuestos y objetivos de desempeño técnicos; frecuente retroalimentación de la compañía patrocinadora y del cliente, así como el compromiso de ambos; y participación del equipo del proyecto en la determinación de las calendarizaciones y los presupuestos, entre otros
- La LDC en su aplicación mediante el uso del *software* Fuzzy Tree Studio provee instrumentos analíticos que están permitiendo integrar una metodología de inteligencia organizacional semántica para la gestión del conocimiento y toma de decisiones. Cualquier problema del lenguaje natural o profesional, se puede simular con estas técnicas, siendo de gran apoyo en la conversión del conocimiento tácito a explícito al mejorar estos procesos en las corporaciones
- Se diseñó un diagrama para la evaluación de proyectos de *software* en Fuzzy Tree Studio, que fue aplicado en la evaluación de tres proyectos
- Se requiere continuar esta investigación mediante la validación de la propuesta por parte de grupos de expertos y ampliar la base de proyectos a evaluar, desarrollando diferentes estudios de caso para, de esta manera, hacer más firme la propuesta

Referencias bibliográficas

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

Baquero Hernández, L. R., Rodríguez Valdés, O., & Ciudad Ricardo, F. Á. (2016). Lógica difusa basada en la experiencia del usuario para medir la usabilidad. *Revista Latinoamericana de Ingeniería de Software, 4*(1), 48-54.

Cejas Montero, J. (2011) La lógica difusa compensatoria. Ingeniería Industrial. XXXII(2). 157-161. Recuperado de: http://www.redalyc.org/pdf/3604/360433576010.pdf

Chao-Ballester, A., & Espín-Andrade, R. A. (2015). Metodología para la gestión del conocimiento y la toma de decisiones basada en lógica difusa compensatoria. En A. Más-Basnuevo, & M. L. Pomín Valentín, *Inteligencia organizacional*

(págs. 163-194). São Paulo: Cultura Académica.

D'Negri, C. E., & De Vito, E. L. (2006). Introducción al razonamiento aproximado: lógica difusa. *Revista Argentina de Medicina Respiratoria*, 4, 126-136. Recuperado de http://www.soneco.com.ar/images/diapositivas/2006-introduccion_al_razonamiento_aproximado-logica_difusa.pdf

Duarte Arias, A. F., & Orozco Ahumada, C. F. (s.f.). Aplicación de un algoritmo genético que incorpora lógica difusa para la minimización del tiempo del proceso de corte de diferentes tipos de materia prima para la empresa Confecciones Taller 84. Tesis de grado.

Espín Andrade, R. A., González Caballero, E., Pedrycz, W., & Fernández González, E. R. (2016). An Interpretable Logical Theory: the Case of Compensatory Fuzzy Logic. *International Journal of Computational Intelligence Systems, 9*(4), 612-626.

----- (2015). Archimedean-compensatory Fuzzy Logic Systems. *International Journal of Computational Intelligence Systems, 8*(2), 54-62.

Espín Andrade, R. A., & Vanti, A. A. (2005). Administración lógica: un estudio de caso en una empresa de comercio exterior. *Revista Base (Administração e Contabilidade) da UNISINOS, 2*(2), 69-77.

Ferreira Lorenzo, G. L., Gálvez Lío, D., Quintero Domínguez, L. A., & Antón Vargas, J. (2014). Estimación del esfuerzo en proyectos de *software* utilizando técnicas de inteligencia artificial. *Revista Cubana de Ciencias Informáticas, 8*(4), 1-20.

Fichtner, C. (2015, 21 de julio). https://twitter.com/pduinsider/status/612869445879508992. Retrieved January 10th, 2016, from https://twitter.com/pduinsider/status/612869445879508992: https://twitter.com/pduinsider/status/612869445879508992

Figueroa, R. G., Solís, C. J., & Cabrera, A. A. (2008). Metodologías tradicionales vs. metodologías ágiles. Universidad Técnica Particular de Loja, Escuela de Ciencias de la Computación.

Gesualdo, S. (2010) Informe Final del Sistema Fuzzy Tree Studio. Mar del Plata : Universidad CAECE.

Grompone, J. (1996). *Gestión de proyectos de* software. Montevideo: La Flor del Itapebí.

Guerrero Ramos, L., Gómez Gutiérrez, E. L., & Armenteros Acosta, M. C. (2014). Mujeres emprendedoras: similitudes y diferencias entre las ciudades de Torreón y Saltillo, Coahuila. *Revista Internacional Administración & Finanzas, 7*(5), 77-90.

Kniberg, H. (2007). *Scrum and XP from the Trenches: How We Do Scrum*. USA: InfoQ Enterprise Software Development Series.

Knorst, A. M., Vanti, A. A., Espín Andrade, R. A., & Johann, S. L. (2011). Aligning Information Security with the Image of the Organization and Prioritization Based on Fuzzy Logic for the Industrial Automation Sector. *Journal of Information Systems and Technology Management, 8*(3), 555-580.

Labra Gayo, J. E., Calvo Salvador, J., Fernández Lanvin, D., & Cernuda del Río, A. (2006). Una experiencia de aprendizaje basado en proyectos utilizando herramientas colaborativas de desarrollo de *software* libre. Oviedo, España: Universidad de Oviedo, Departamento de Informática.

Letelier Torres, P., & Penadés, M. C. (2006). Metodologías ágiles para el desarrollo de *software*: eXtreme Programming (XP). *Revista Ciencia y Técnica Administrativa (CYTA), 5*(26).

López Martín, C. (2017). Modelo para la estimación del esfuerzo de desarrollo en proyectos de *software* a nivel personal aplicando lógica difusa. *Dspace Repository*.

Markus, M. (2010). Failed Software Projects? Not Anymore. *Quality Progress*, 116-117.

McManus, J., & Wood-Harper, T. (2003). Information Systems Project Management: the Price of Failure, 16-19.

Mendoza, L.E., Pérez, M. A., & Grimán, A.C. (2005) Prototipo de modelo sistémico de calidad (MOSCA) del software. Computación y Sistemas. 8(3). 196-217.

Meschino, G., Nabte, M., Gesualdo, S., Monjeau, J. A., & Passoni, I. (2013) Fuzzy Tree Studio: a tool for the design of the Scorecard for the Management of Protected Areas. En: Soft Computing for Bussiness Intelligence. Berlin: Springer Verlag. 99-112.

Peña González, M., Barrionuevo de la Rosa, C. G., & Cedeño Morán, F. J. (2016). Grupo focal y procesos de jerarquía analítica para la determinación y ponderación de los factores críticos de éxitos en los proyectos de *software*. *International Journal of Innovation and Applied Studies, 15*(4), 743-746.

Piattini, M. G., & García, F. O. (2003). *Calidad en el desarrollo y mantenimiento del* software. Alfaomega.

Pinto, J. K., & Mantel, Jr., S. J.(1990). The Causes of Project Failure. *IEEE Transactions of Engineering Management, 37*(4), 269-276.

Pressman, R. S. (2002). *Ingeniería del* software*: un enfoque práctico.* McGraw-Hill Interamericana.

Remón, C. A., & Thomas, P. (2010). Análisis de estimación de esfuerzo aplicando puntos de caso de uso. XVI Congreso Argentino de Ciencias de la Computación.

Rodríguez Bello, S. (2008). Toma de decisión multicriteria con AHP, ANP y lógica difusa. Universidad Nacional de Colombia.

Rodríguez, A., Ortega, F., & Concepción, R. (2016). A Method for the Evaluation of Risk in IT Projects. *Experts Systems with Applications, 45*(1), 273-285.

Schwaber, K. (2004). *Agile Project Management with Scrum*. Microsoft.

Sommerville, I. (2000). *Ingeniería del* software (7.ª ed., M. I. Alfonso Galipienso, A. Botía Martínez, F. Mora Lizán, & J. P. Trigueros Jover, Trads.) Alicante: Pearson Addison Welsey.

Standish Group (2014). The Standish Group Report: "Chaos". Recuperado de https://www.projectsmart.co.uk/white-papers/chaos-report.pdf

Vanti, A. A., & Espín Andrade, R. A. (2007). Metodologia multivalente para priorizaçao estrategica em construçao de Balanced Scorecard. *Revista do Centro de Ciências da Economia e Informática (CCEI), 11*(20), 54-67.

Velarde Bedregal, H. R., & Santiesteban, C. (2017). Modelo para la estimación del esfuerzo de desarrollo en tareas de ingeniería de proyectos de *software* empleando aprendizaje automático.

Weston, A. (2006). *Las claves de la argumentación.* Barcelona: Ariel Letras.

Zapata Jaramillo, C. M., & Arango Isaza, F. (2005). Los modelos verbales en lenguaje natural y su utilización en la elaboración de esquemas conceptuales para el desarrollo de *software*: una revisión crítica. *Revista Universidad EAFIT, 41*(137), 77-95.

Zavala Ruiz, J. (2014). ¿Por qué fracasan los proyectos de *software*?; un enfoque organizacional. Congreso Nacional de Software Libre 2004.