Universidad de las Ciencias Informáticas

Facultad 3

Herramienta de software para la toma de decisiones en los análisis de factibilidad

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor:

Roniel López Álvarez

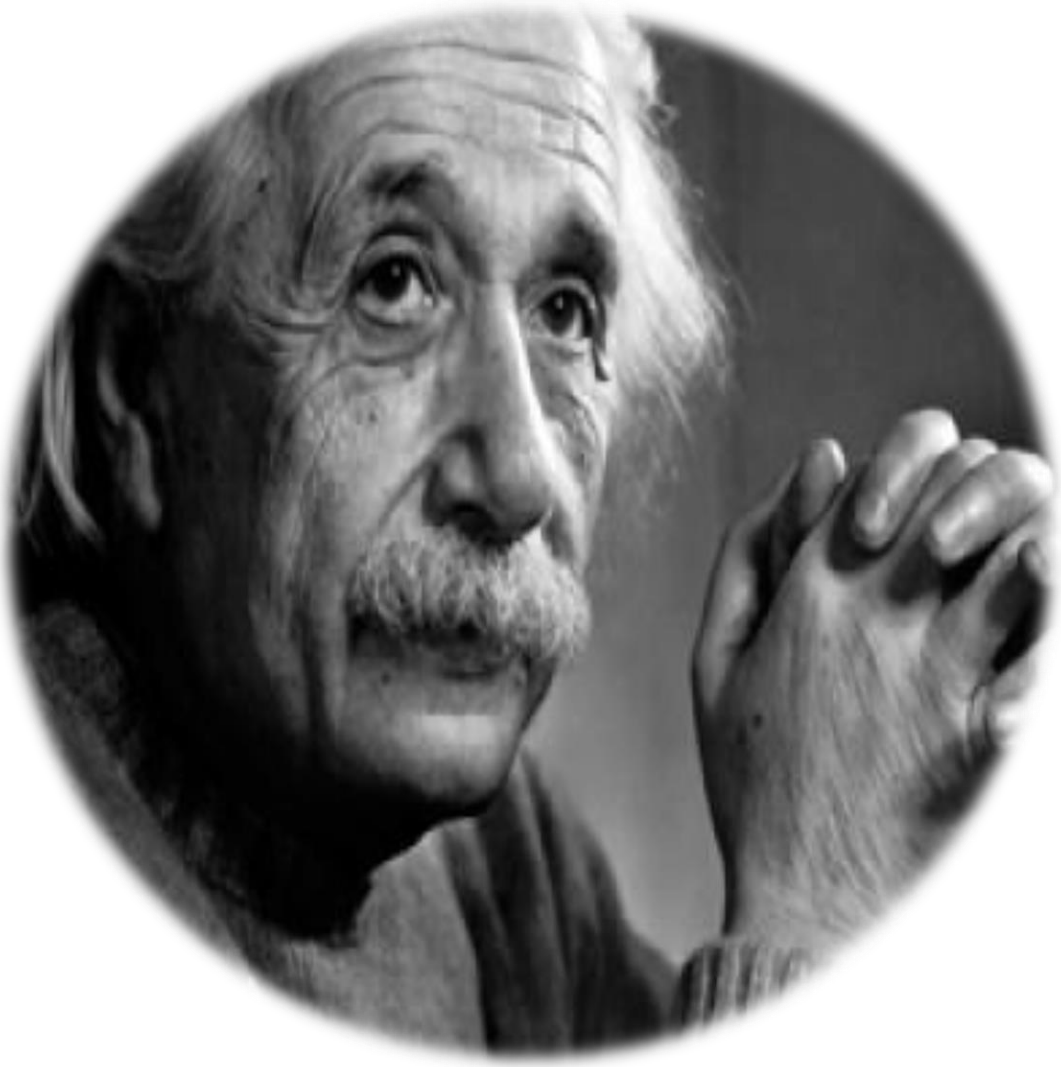
Tutores:

DrC. Marieta Peña Abreu

M.Sc. Carlos Rafael Rodríguez Rodríguez

La Habana, julio del 2018

"Año 60 de la Revolución"



LA MEDIDA DE LA INTELIGENCIA ES LA CAPACIDAD DE CAMBIAR

ALBERT EINSTEIN

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Roniel López Álvarez

_____

Firma del Autor

DrC. Marieta Peña Abreu

_____

Firma del Tutor

M.Sc. Carlos Rafael Rodríguez Rodríguez

_____

Firma del Tutor

Agradecimientos

A mi madre Yaritza, por el apoyo incondicional que siempre me ha dado.

A mis abuelos (Rosendo y Andrea), siempre a mi lado en todo momento, respaldando cada idea y cada sueño.

A mi hermana Niuris que, a pesar de sus locuras, siempre tiene tiempo para estar al tanto de mis problemas.

A todos aquellos con los que compartí mi tiempo y mi vida en la Universidad, y que indudablemente han contribuido a formar la persona que soy hoy.

A mi esposa, por enseñarme un mundo nuevo y darme el regalo más hermoso que pueda existir, un pedacito de mí que pronto tendré en mis brazos y que desde ya es la fuerza que me impulsa a ser mejor profesional y mejor persona.

A mis tutores Marieta y Carlos, por su dedicación y ayuda.

A todos mis profesores durante la carrera. Ellos hicieron posible mi formación como persona e ingeniero.

DEDICATORIA

A mi hija que desde ya es la fuerza que me impulsa a ser mejor profesional y mejor persona.

RESUMEN

Los análisis de factibilidad contribuyen a la toma decisiones en las organizaciones al decidir la ejecución o no del proyecto, ya que evalúan la disponibilidad de recursos para enfrentar el desarrollo y los beneficios que puede obtenerse. Los proyectos informáticos imponen retos adicionales dado el carácter intangible del software y su asociación al desarrollo de conocimientos. Las principales escuelas de Gestión de Proyectos suponen que se haya realizado con anterioridad un análisis de factibilidad y no brindan métodos para su realización. Para resolver éstas limitaciones los análisis de factibilidad pudieran tratarse como un problema de toma de decisiones. Desde sus inicios los análisis de factibilidad estuvieron muy relacionados con el cálculo de criterios económicos completamente deterministas, sin tener en cuenta la incertidumbre inherente en el ambiente de desarrollo y otro conjunto de criterios como los técnicos, comerciales y sociales. El presente trabajo tiene como objetivo desarrollar una herramienta informática para el análisis de factibilidad de proyectos de software, usando técnicas de softcomputing que contribuya a la toma de decisiones con una mayor predicción de la factibilidad.

La herramienta fue desarrollada utilizando herramientas y tecnología libres, como metodología de desarrollo se empleó Programación Extrema (XP), el marco de trabajo Django para el trabajo con el lenguaje de programación Python. La solución desarrollada contribuye al análisis de factibilidad de proyectos de software para la toma de decisiones con tratamiento de la incertidumbre.

Palabras claves: criterios, estudios de factibilidad, proyectos, incertidumbre, toma de decisiones

ÍNDICE

ÍNDICE

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La creciente utilización de las Tecnologías de la Información y las Comunicaciones (TIC) y su impacto en la sociedad conllevan a que las organizaciones se enfrenten a mayores retos en la obtención de proyectos exitosos. La selección adecuada de un proyecto a desarrollar puede impulsar la economía y la sociedad (1), pero es una tarea compleja, ya que está dada por múltiples factores que se deben considerar para tomar tal decisión. Los análisis de factibilidad contribuyen en la toma decisiones en las organizaciones al decidir la ejecución o no del proyecto, ya que evalúan la disponibilidad de recursos para enfrentar el desarrollo, así como los beneficios que puede retornar (2).

Un componente que ha impulsado el desarrollo social, es el despliegue de productos de software, que inciden en la productividad, la planificación, el control, la calidad de los procesos, entre otros disímiles aspectos. Los proyectos informáticos imponen retos adicionales dado el carácter intangible del software y su asociación al desarrollo de conocimientos (3). A pesar de los esfuerzos realizados por las organizaciones, estudios publicados por Standish Group, en el Chaos Manifiesto 2015 (4) alrededor de 50 000 proyectos, arrojó que el 52% de éstos fue renegociado y el 19% fallido, siendo una de las principales causas el aumento de los costos durante su ejecución.

Las principales escuelas de Gestión de Proyectos (ISO 21500, PMBOK, CMMI, IPMA, PRINCE2) suponen que se haya realizado con anterioridad el análisis de factibilidad y no brindan métodos para su realización. Para resolver éstas limitaciones los análisis de la factibilidad pudieran tratarse como un problema de toma de decisiones (5). A partir de la analogía con este tipo de problemas, se deben considerar elementos tales como: el ambiente de desarrollo, los criterios a evaluar, los participantes y los métodos a utilizar.

Dentro de los criterios a evaluar, desde sus inicios los análisis de factibilidad estuvieron muy relacionados con el cálculo de criterios económicos completamente deterministas, sin tener en cuenta la incertidumbre inherente en el ambiente de desarrollo (3) (6). La incertidumbre en el proceso de solución de problemas puede estar presente tanto en los datos de la instancia del problema que se resuelve, como en el conocimiento que se utiliza para resolverlo, en este caso está presente en ambos. Esto implica que en este contexto sea necesario modificar la manera de concebir las herramientas lo cual se viene infiriendo desde hace años (7) (8).

Además de los criterios económicos en los análisis de factibilidad, se proponen evaluar otros tales como: técnicos, comerciales, sociales y medioambientales, estrechamente relacionados con la sostenibilidad del proyecto (9) (10) (11). Para su cálculo por lo general se utilizan métodos tradicionales de toma de decisiones (12) (13), dentro de los más utilizados los de evaluación multicriterios.

La gran mayoría de las herramientas existentes a nivel mundial (14) (15) (16) (17) (18) para la evaluación de proyectos son privativas, no evalúan en conjunto las aristas económica, técnica, comercial, social y medioambiental de un proyecto. No tienen en cuenta la incertidumbre presente en la información disponible sobre las distintas alternativas a evaluar, la cuál puede ser incompleta, vaga o imprecisa, lo que implica que la evaluación asignada a cada alternativa tenga que ser valorada de forma cualitativa.

Para manejar la incertidumbre, los investigadores han utilizado diferentes técnicas de softcomputing según el contexto (19) (20) (21) (22). Entre los paradigmas que más se destacan se encuentra la computación con palabras (CWW, del inglés Computing With Words) con éxito para resolver problemas similares en otras áreas.

A partir de lo mencionado anteriormente, queda definido como **problema a resolver:** las insuficiencias en las herramientas informáticas para el análisis de factibilidad de proyectos de software, están afectando la toma de decisiones.

Para dar solución al problema descrito se define como **objetivo general**: desarrollar una herramienta informática para el análisis de factibilidad de proyectos de software, usando técnicas de softcomputing que contribuya a la toma de decisiones.

Se determina como **objeto estudio:** análisis de factibilidad de proyectos de software.

Delimitándose en el **campo de acción:** herramientas informáticas para el análisis de factibilidad de proyectos de software para la toma de decisiones.

Como **idea a defender** se plantea: el desarrollo de una herramienta informática para el análisis de factibilidad de proyectos de software usando técnicas de softcomputing, contribuirá a mejorar la toma de decisiones.

Para dar cumplimiento al objetivo general se definen como **objetivos específicos:**

- Construir el marco teórico de la investigación relacionado con las herramientas para el análisis de factibilidad de proyectos de software, así como metodologías, herramientas y lenguajes para el desarrollo de software.
- Desarrollar los artefactos ingenieriles para la obtención de la herramienta informática.
- Comprobar la validez de la herramienta a partir de la validación de las variables dependientes y el correcto funcionamiento del software.

A continuación, se presentan los **métodos de investigación** utilizados:

Métodos teóricos

- Histórico-lógico: se emplea para el estudio de las herramientas existentes para el análisis de factibilidad de proyectos de software.
- Analítico-sintético: se emplea para realizar el estudio teórico de la investigación, haciendo posible la selección de los elementos más importantes en el proceso de desarrollo de la herramienta.
- Inductivo-deductivo: se emplea para el razonamiento de la información consultada y llegar así a la obtención de un grupo de conocimientos particulares y generales.
- Modelación: se emplea para en junto con la metodología de desarrollo seleccionada, modelar los diagramas esenciales para el desarrollo de la herramienta.

Métodos empíricos

- Entrevista: se emplea en el intercambio con el cliente para adquirir información necesaria sobre el negocio, así como los requisitos que se deben cumplir en el desarrollo de la herramienta.

El presente documento está compuesto por 3 capítulos, estructurados de la siguiente manera:

Capítulo 1: Fundamentación teórica, se realiza una breve explicación acerca de los análisis de factibilidad. Se lleva a cabo una revisión y análisis de las herramientas existentes para el análisis de factibilidad de proyectos de software. Se plantea la técnica de softcomputing a utilizar, computación con palabras, y dentro de esta el modelo lingüístico 2-tuplas. Se selecciona la metodología de desarrollo de software a emplear,

los lenguajes, herramientas de desarrollo y patrones a usar para dar solución al problema planteado. Por último, se describen las pruebas a realizar para la validación y verificación del sistema.

Capítulo 2: Propuesta de solución, se propone la solución de la investigación. Se exponen los requisitos funcionales y no funcionales. Se modelan los artefactos correspondientes a las fases de Planificación y Diseño de la solución. Se describen los patrones de diseño y arquitectónicos empleados.

Capítulo 3: Validación de la solución, se obtiene los artefactos correspondientes a la fase de Desarrollo y Prueba. Se evalúa el cumplimiento de los objetivos planteados a partir de la realización de pruebas al sistema, pruebas de aceptación por parte del cliente y la validación de la herramienta desarrollada a partir de la validación de la solución.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción al capítulo

En este capítulo se define la fundamentación teórica que rige la investigación acerca de los análisis de factibilidad y las diferentes herramientas existentes para su aplicación con vista a la toma de decisiones. Se plantea la técnica de softcomputing a utilizar, computación con palabras, y dentro de esta el modelo lingüístico 2-tuplas. Se describen las principales tecnologías, lenguajes de programación y herramientas definidas para el desarrollo de la herramienta, así como la metodología de desarrollo de software, los patrones a emplear, las métricas para la validación del diseño y las pruebas para garantizar la calidad del producto final.

1.1 Análisis de factibilidad de proyectos

Existen disímiles definiciones asociadas a los análisis de factibilidad, Salazar plantea que estos son los análisis que se realizan para conocer si los proyectos son o no técnica, financiera, económica, social, ambiental y jurídicamente viables (23). Pressman establece cuatro dimensiones para la factibilidad: tecnología, finanzas, tiempo y recursos (24). Por otro lado, Sommerville puntualiza que los análisis consideran si el sistema propuesto tendrá un costo-beneficio desde un punto de vista empresarial, y si éste puede desarrollarse dentro de las restricciones presupuestales existentes. Debe ser rápido y relativamente barato. El resultado debe informar la decisión respecto a si se continúa o no continúa con un análisis más detallado (25).

Los análisis de factibilidad reducen la incertidumbre y mejoran la calidad de la información al ahondar los estudios de mercado, técnico y financiero y, en función de los resultados, seleccionar la alternativa óptima. De no existir alternativas rentables el proyecto es descartado (26). Se debe verificar la existencia de un mercado potencial o de una necesidad no satisfecha. Demostrar la viabilidad técnica y la disponibilidad de los recursos humanos, materiales, administrativos y financieros (27).

A continuación, se relacionan conceptualmente los diferentes tipos de análisis de factibilidad a tener en cuenta (27):

Factibilidad técnica: hace referencia a la viabilidad desde el punto de vista técnico para adelantar un proyecto.

Factibilidad financiera: considera los beneficios y costos que puede percibir un inversionista privado, a partir de los precios de mercado.

Factibilidad económica: se realiza para garantizar una asignación óptima de los recursos económicos disponibles, y el logro de los objetivos propuestos; teniendo en cuenta el costo de los insumos y la magnitud del impacto que produce en el medio económico donde se inserta.

Factibilidad social: se orienta a medir los efectos de un proyecto sobre la sociedad en su conjunto. Considera todos los beneficios y costos que puedan afectar a la sociedad. Trata de identificar los efectos del proyecto sobre la distribución del ingreso, por lo que se considera como un indicador de equidad.

Factibilidad comercial: encargado de estudiar el mercado a través de la información, la cual se utiliza para identificar y definir tanto las oportunidades como las amenazas del entorno. Por su carácter preliminar, constituye un sondeo de mercado, antes de incurrir en costos innecesarios (28).

1.2 Herramientas existentes para el análisis de factibilidad de proyectos

A continuación, se relacionan las herramientas estudiadas para el análisis de factibilidad de proyectos de software y apoyo a la toma de decisiones.

1.2.1 EasyPlanEx

Software que provee una solución integral para evaluar y optimizar la formulación de proyectos de inversión, desarrollado por BoraSystems. Permite analizar todas las opciones posibles para el proyecto y encontrar automáticamente la mejor formulación. Determina la tasa de descuento como costo promedio ponderado de capital y calcula el costo del capital. Realiza análisis de sensibilidad unidimensional y multidimensional. Mide el riesgo del proyecto, considerando la incertidumbre en relación con las variables que no se controlan, mediante la técnica de Montecarlo[1]. Genera automáticamente una documentación completa y consistente del proyecto, que refleja el modelo desarrollado y describe los supuestos y datos empleados (14).

La herramienta **EasyPlanEx** es privativa por lo que tiene costo de licencia para su uso, evalúa fundamentalmente un proyecto desde la arista económica, tiene en cuenta la incertidumbre en la medición de los riesgos.

[1] La técnica de Montecarlo es una técnica numérica para calcular probabilidades y otras cantidades relacionadas, utilizando secuencias de números aleatorios.

1.2.2 EvalAs

Software de evaluación de proyectos. Permite el ingreso de información de proyectos como: inversiones, costos fijos, costos variables e impuestos. A partir de los datos ingresados, el sistema calcula la Matriz de Flujos de Caja y los principales indicadores financieros: VAN, Período de Repago, TIR, Período de Repago con Descuento. Realiza una simulación de Montecarlo de cualquier ítem del proyecto. Se puede realizar análisis de sensibilidad y de escenarios con manejo de la incertidumbre. Permite la comparación con otro proyecto para determinar cuál alternativa es la más rentable (17).

La herramienta **EvalAs** es privativa, evalúa un proyecto desde la arista económica, tiene en cuenta la incertidumbre en el análisis de sensibilidad y de escenarios.

1.2.3 Intecplan

Software para elaborar un Proyecto de Inversión o Plan de Negocios competitivo. Permite realizar estudios de mercado, técnico, financiero y de organización. Cumple los requisitos técnicos de los programas de financiamiento, TIR, VPN, TR, punto de equilibrio, rendimiento. Organiza tablas de Excel prediseñadas y archivos de Word que siguen una secuencia que incluye los estudios y capítulos descriptivos del proyecto (18).

La herramienta **Intecplan** es privativa, evalúa un proyecto desde las aristas comercial, técnica y económica, no realiza tratamiento de la incertidumbre.

1.2.4 Expert Choice

Software para la toma de decisiones multicriterio, basado en el Proceso Jerárquico Analítico (AHP, Analytic Hierarchy Process). Ha sido usado exitosamente en una variedad de aplicaciones incluyendo, priorización y evaluación de proyectos, planeamiento estratégico y análisis de costo/beneficio. Cuenta con una amplia gama de aplicaciones entre las que se encuentran: Asignación de Recursos, Evaluación de personal, Formulación de estrategias de marketing, Administración de la producción, Evaluación de proveedores, Facilitación de decisiones grupales. Ofrece soluciones basadas en la nube para que los equipos distribuidos puedan resolver de forma colaborativa problemas complejos con un proceso de toma de decisiones estructurado, repetible y justificable. (15).

La herramienta **Expert Choice** es privativa, evalúa un proyecto desde las aristas comercial y económica, no realiza tratamiento de la incertidumbre.

1.2.5 Decide

Software especializado que facilita la elaboración de planes de negocios, formulación y evaluación de proyectos de inversión, proporciona metodologías y herramientas para una toma de decisiones sustentada. Desarrollado por la empresa mexicana Soluciones Informáticas y Aplicaciones Crediticias S. A. de C.V. Permite realizar evaluación de un proyecto de inversión, elaborar estudios de mercado, análisis técnicos, económicos, financieros y de riesgos (16).

La herramienta **Decide** es privativa, evalúa un proyecto desde las aristas comercial, técnica y económica, no realiza tratamiento de la incertidumbre.

1.2.6 Gespro

Paquete de Gestión de Proyectos desarrollado en la Universidad de las Ciencias Informáticas (UCI), registrado en el Centro Nacional de Derecho de Autor (CENDA) con el No. de Registro 1540-2010. Abarca varias áreas de la Gestión de Proyectos, las cuales están presentes en el sistema mediante funcionalidades y módulos. Entre estas se encuentran: la gestión de portafolios de proyectos, la gestión del alcance de productos, la gestión del tiempo, la gestión de riesgos de proyectos, la gestión de comunicaciones, la gestión de recursos humanos y materiales y la gestión documental. Es una herramienta libre de costo.

La suite Gespro incluye la implementación del modelo MFac-PS, el cual realiza el análisis de factibilidad de proyectos de software en entornos de incertidumbre, a partir de la evaluación de criterios económicos, técnicos, comerciales y sociales utilizando técnicas de softcomputing. Este modelo corresponde a la tesis de doctorado de la DrC. Marieta Peña Abreu.

Valoración de las herramientas

Luego de realizado el estudio de las herramientas anteriores se muestra una tabla comparativa entre ellas, atendiendo a diferentes criterios:

- tratamiento de la incertidumbre
- análisis de diferentes aristas (económica, comercial, social, técnica)
- licencia

Tabla 1. Comparación de las herramientas

Herramientas	Tratamiento de la incertidumbre	Aristas	Licencia
EasyPlanEx	sí	económica	privativa
EvalAs	sí	económica	privativa
Intecplan	no	comercial, técnica y económica	privativa
Expert Choice	no	comercial y económica	privativa
Decide	no	comercial, técnica y económica	privativa
Gespro	sí	económica, técnica, comercial y social	libre

Las herramientas analizadas facilitan la toma de decisiones luego de realizado el análisis de factibilidad de un proyecto. Como se evidencia en la tabla de las 6 estudiadas 5 son privativas, por lo que es necesario pagar una licencia para su uso. En su mayoría están centrada en la evaluación de criterios económicos, algunas tienen en cuenta además criterios comerciales y técnicos, y solo una evalúa en su conjunto todos los criterios anteriores incluyendo el social. En el caso de Gespro realiza el análisis de todos los indicadores mencionados con tratamiento de la incertidumbre, mediante el modelo MFac-PS como soporte computacional asociado a la herramienta. Como se mencionaba anteriormente Gespro es una suite para la gestión de proyectos por lo que incluye además de los análisis de factibilidad áreas como la gestión del alcance, la gestión del tiempo, la gestión de riesgos, entre otras. Para empresas o entidades que solo soliciten un sistema para el análisis de factibilidad de sus proyectos, teniendo en cuenta la evaluación económica, técnica, comercial y social con tratamiento de la incertidumbre, las herramientas anteriores no satisfacen completamente sus necesidades. Por lo anteriormente expuesto en el presente trabajo se propone el desarrollo de una herramienta informática para el análisis de factibilidad de proyectos de software, usando técnicas de softcomputing para el tratamiento de la incertidumbre que contribuya a la toma de decisiones.

1.3 Técnica de softcomputing para el tratamiento de la incertidumbre

1.3.1 Computación con palabras

La computación con palabras, en lo adelante CWW[2], es una técnica de softcomputing en la cual se consideran palabras y proposiciones del lenguaje natural como los principales objetos de computo. Persigue la habilidad humana de resolver tareas y tomar decisiones en escenarios con imprecisión en la información. Contribuye al razonamiento y la toma de decisiones (29). La ***Figura 1*** muestra diferentes enfoques para el tratamiento de información heterogénea, resaltando el modelo de representación 2-tuplas.

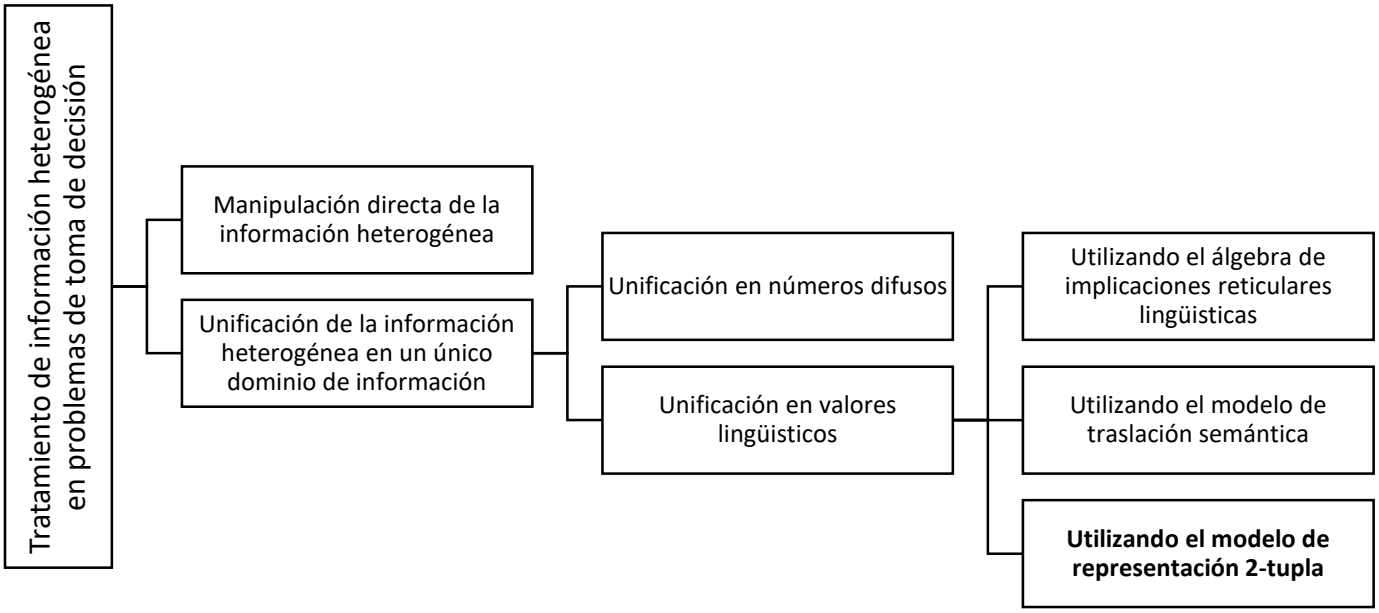


Figura 1. Clasificación de enfoques para el tratamiento de información heterogénea

1.3.2 Modelo lingüístico 2-tuplas

Dada la naturaleza heterogénea de las evaluaciones emitidas por los expertos, es necesario transformarlas a un único dominio para que exista un marco común de expresión para operar sobre ellas. Para llevar a cabo lo antes planteado se utiliza el dominio lingüístico, siguiendo lo propuesto por Herrera (30) y se utiliza un Conjunto Básico de Términos Lingüísticos (CBTL) de siete etiquetas, definido en (31) y representado en

[2] **CWW:** Acrónimo en inglés de Computing With Words

la **Figura *2***, que en lo adelante se representará como S_T. Cada evaluación dada en el dominio numérico, intervalar o lingüístico se expresa mediante un conjunto difuso en S_T, $F(S_T)$.

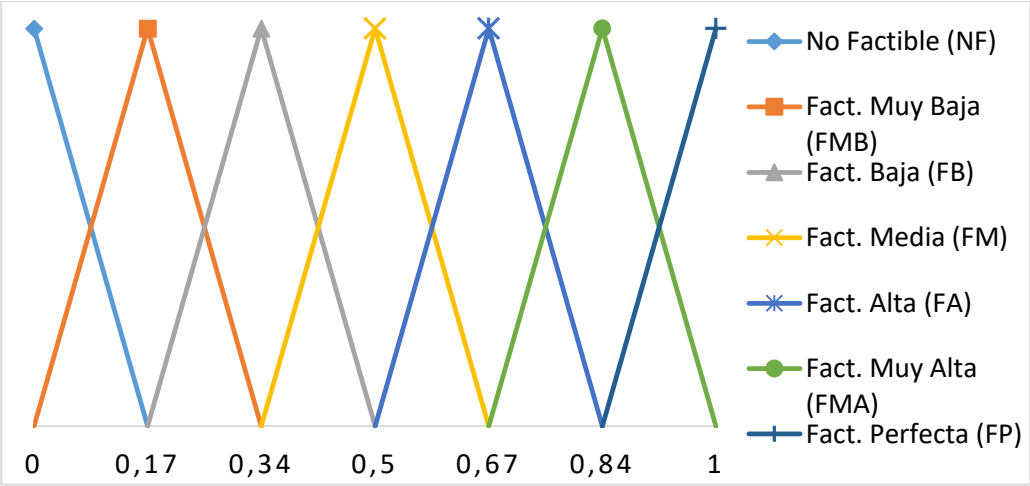


Figura 2. Conjunto básico de términos lingüísticos de siete etiquetas con su semántica asociada

Para llevar a cabo las transformaciones se emplean las definiciones dadas por Herrera (30) que se formalizan como sigue:

a) para transformar del dominio numérico al lingüístico (T_{NS_T}):

$$\boldsymbol{T_{NS_T}}: [\mathbf{0}, \mathbf{1}] \rightarrow \boldsymbol{F(S_T)} \qquad (1)$$

$$\boldsymbol{T_{NS_T}}\ (\boldsymbol{N}) = \{(\boldsymbol{S_0}, \boldsymbol{\gamma_0}), \dots, (\boldsymbol{S_g}, \boldsymbol{\gamma_g})\}, \boldsymbol{S_i} \in \boldsymbol{S_T}\ y\ \boldsymbol{\gamma_i} \in [\mathbf{0}, \mathbf{1}] \qquad (1)$$

$$\boldsymbol{\gamma_i} = \boldsymbol{\mu_{S_i}(N)} = \begin{cases} \mathbf{0}, & \boldsymbol{N < a}\ \text{ó}\ \boldsymbol{N > c} \\ \dfrac{\boldsymbol{N - a}}{\boldsymbol{b - a}}, & \boldsymbol{a < N < b} \\ \mathbf{1}, & \boldsymbol{N} = b \\ \dfrac{\boldsymbol{c - N}}{\boldsymbol{c - b}}, & \boldsymbol{b < N < c} \end{cases} \qquad (2)$$

b) para transformar la información intervalar al dominio lingüístico (T_{IS_T}):

$$\boldsymbol{T_{IS_T}}: \boldsymbol{I} \rightarrow \boldsymbol{F(S_T)} \qquad (3)$$

$$\boldsymbol{T_{IS_T}}\ (\boldsymbol{I}) = \left\{\left(\boldsymbol{S_k}, \boldsymbol{\gamma_k^i}\right)/\boldsymbol{k} \in \{\mathbf{0}, \dots, \boldsymbol{g}\}\right\} \qquad (4)$$

$$\boldsymbol{\gamma_k^i} = \boldsymbol{max_y min}\{\boldsymbol{\mu_I(y)}, \boldsymbol{\mu_{S_k}(y)}\} \qquad (5)$$

Donde $F(S_T)$ es el conjunto definido en $S_T = \{S_0, \dots, S_g\}$, y $\mu_I(\cdot)$ y $\mu_{S_k}(\cdot)$ son las funciones de pertenencia asociadas con el intervalo I y el termino S_k respectivamente. Dado un intervalo $[a, b]$, su función de pertenencia se define como sigue:

$$\mu_I(x) = \begin{cases} 0, & x < a \\ 1, & a \le x \le b \\ 0, & x > b \end{cases} \qquad (6)$$

c) para transformar la información lingüística al dominio lingüístico seleccionado (T_{SS_T}):

$$T_{SS_T}: S \to F(S_T) \qquad (7)$$

$$T_{SS_T}(l_i) = \left\{\left(S_k, \gamma_k^i\right)/k \in \{0, \dots, g\}\right\} \ \forall\, l_i \in S \qquad (8)$$

$$\gamma_k^i = max_y min\{\mu_{l_i}(y), \mu_{S_k}(y)\} \qquad (9)$$

Donde $S_T = \{S_0, \dots, S_g\}$ y $S = \{l_0, \dots, l_p\}$ son dos conjuntos difusos tal que $g \ge p$ y $\mu_{l_i}(\cdot)$ y $\mu_{S_k}(\cdot)$ son las funciones de pertenencia asociadas con los términos l_i y S_k respectivamente.

Luego de tener todas las evaluaciones en un mismo dominio, se transforman a 2-tuplas para facilitar el proceso de ordenado en la fase de explotación de los resultados y para ello se utiliza la función de transformación definida por Herrera (30) que se muestra a continuación:

$$X(F(S_T)) = \Delta\left(\frac{\sum_{j=o}^{g} j\gamma_j}{\sum_{j=o}^{g} \gamma_j}\right) = \Delta(\beta) = (S, \alpha) \qquad (10)$$

$$Donde\, \Delta(\beta) = \begin{cases} Si, & i = round(\beta) \\ \alpha = \beta - i, & \alpha \in [-0.5, 0.5] \end{cases} \qquad (11)$$

Teniendo todas las evaluaciones transformadas a 2-tuplas se agregan las preferencias de todos los expertos para obtener el valor colectivo de cada criterio para cada proyecto evaluado. Se utiliza el operador Media Aritmética Extendida (32), que significa el punto de equilibrio del conjunto de valores y que se formaliza en como sigue:

$$\Phi(x) = \Delta\left(\frac{1}{m}\sum_{i=1}^{m} \Delta^{-1}((S_i, \alpha_i))\right) = \Delta\left(\frac{1}{m}\sum_{i=1}^{m} \beta_1\right) \qquad (12)$$

Como resultado se obtiene la 2-tupla lingüística que simboliza el valor medio de cada criterio para cada proyecto y la precisión de esa información.

Para agregar el valor de los criterios de cada proyecto se considerando su peso. Se utiliza el operador Media Ponderada Extendida (32). La agregación mediante este operador se lleva a cabo mediante la fórmula:

$$\Psi(x) = \Delta\left(\frac{\sum_{i=1}^{m} w_i \Delta^{-1}((s_i,\alpha_i))}{\sum_{i=1}^{m} w_i}\right) = \Delta\left(\frac{\sum_{i=1}^{m} w_i \beta_i}{\sum_{i=1}^{m} w_i}\right) \qquad (13)$$

Como resultado de este paso se obtiene la 2-tupla que representa el valor de factibilidad de cada proyecto y la precisión de esa información. La 2-tupla que representa la factibilidad de cada proyecto, contiene el término lingüístico correspondiente y la precisión de esa evaluación. La precisión es un valor entre [-0.5, 0.5] que indica la distancia entre el punto de máxima pertenencia al termino lingüístico y el resultado de la agregación. Utilizando el valor de precisión es posible determinar la certeza de la factibilidad calculada para cada proyecto.

Para analizar los resultados de evaluar en conjunto a todos los proyectos y obtener un listado ordenado de los mismos en función de su factibilidad, se deben emplear los operadores de comparación para 2-tuplas (32) que plantean que para dos 2-tuplas (s_k, α_1) y (s_l, α_2) que representan dos evaluaciones:

- Si $k > l$ entonces $(s_k, \alpha_1) > (s_l, \alpha_2)$
- Si $k < l$ entonces $(s_k, \alpha_1) < (s_l, \alpha_2)$
- Si $k = l$ entonces:
  - Si $\alpha_1 = \alpha_2$ entonces $(s_k, \alpha_1) = (s_l, \alpha_2)$
  - Si $\alpha_1 < \alpha_2$ entonces $(s_k, \alpha_1) < (s_l, \alpha_2)$
  - Si $\alpha_1 > \alpha_2$ entonces $(s_k, \alpha_1) > (s_l, \alpha_2)$

A partir de este momento se pueden tomar alguna de las siguientes decisiones:

- Seleccionar los proyectos considerando solo su factibilidad total (global).
- Si dos proyectos tienen el mismo resultado de factibilidad global, podrá aumentarse el nivel de detalles ordenándolos según alguno de los criterios analizados.
- Es posible decidir sobre cada proyecto de manera individual, basando el análisis para cada uno, en los criterios más relevantes según su naturaleza y considerando además las condiciones actuales y planes futuros de la organización desarrolladora.
- En caso de que haya un único proyecto se debe tomar la decisión de si es factible su desarrollo o no.

1.4 Metodología de desarrollo

Dentro del desarrollo de software y con la necesidad de que los proyectos lleguen al éxito y obtener un producto de gran valor para los clientes, se hace necesario el empleo de una metodología. La metodología seleccionada debe ser aquella que mejor se ajuste a las características del equipo de desarrollo y las exigencias de los usuarios finales.

Las metodologías de desarrollo se pueden enmarcar en dos grandes grupos, las metodologías tradicionales y las metodologías ágiles. Las tradicionales enfatizan en el uso exhaustivo de documentación durante todo el ciclo de vida del proyecto. Son recomendadas para proyectos de grandes dimensiones y con grandes equipos de desarrollo. Las metodologías ágiles dan importancia a la capacidad de respuesta a los cambios, se enfatiza en la satisfacción del cliente y promueven el trabajo en equipo (33).

Para poder llevar a cabo el desarrollo de la propuesta de solución en un corto período de tiempo, donde el cliente esté en constante participación y colaboración y poder realizar cambios en los requisitos si fuese necesario, se opta por una metodología ágil de desarrollo de software en lugar de una metodología tradicional.

Se realizó un estudio de las metodologías ágiles XP, SCRUM y AUP, teniendo en cuenta la respuesta a los cambios, el trabajo con el cliente, los planes de entrega de las iteraciones, el trabajo en equipo y el enfoque.

La **Programación Extrema** (*e**X**treme **P**rogramming*, XP) es posiblemente el método ágil más conocido y ampliamente utilizado. En esta las iteraciones de entrega son entre una y tres semanas. Cuando se van finalizando las entregas son susceptibles a modificaciones durante el transcurso de todo el proyecto. El cliente forma parte del equipo de desarrollo, está en constante participación. Los miembros programan en parejas. Se centra más en la propia programación o creación del producto.

La metodología **SCRUM** está enfocada a la gestión de procesos. En esta las iteraciones de entregas son de dos a cuatro semanas. Al finalizar una entrega las tareas que se han realizado no pueden ser modificadas. Se mantiene una estrecha colaboración con el cliente. Cada miembro del equipo trabaja de forma individual. Se centra más en la administración del proyecto.

En la metodología **AUP** la primera producción de liberación puede tomar doce meses, para entregar la segunda versión nueve meses, y luego otras liberaciones se entregan cada seis meses. Permite una gestión de cambios ágil. Se mantiene una estrecha colaboración con el cliente. Se centra especialmente en la gestión de riesgos. Propone que aquellos elementos con alto riesgo obtengan prioridad en el proceso de desarrollo y sean abordados en etapas tempranas del mismo.

Se selecciona como metodología de desarrollo la **Programación Extrema (XP)** por ser una metodología flexible a los cambios en las entregas, diseñada para equipos de trabajo pequeños donde la programación es por parejas, lo que cumple con las características del equipo de desarrollo. El cliente será partícipe del proceso de desarrollo estando en constante participación. Por parte del cliente se solicita que las entregas por iteraciones sean de un corto período de tiempo, con lo cual cumple esta metodología. Lo más importante tanto para el equipo de desarrollo como para el cliente es el producto final. Se generan los artefactos mínimos para la comunicación con el cliente. Esta consta de 4 fases: planificación, diseño, desarrollo y pruebas.

1.5 Herramientas, tecnologías y lenguajes

Se seleccionó como marco de trabajo Django en su versión 2.0.5 ya que el desarrollador tiene experiencia en el desarrollo de aplicaciones usando este framework, además, provee un alto nivel de abstracción de patrones comunes en el desarrollo web, atajos para tareas frecuentes de programación y convenciones claras sobre cómo solucionar problemas. Django permite crear y mantener aplicaciones web de alta calidad con un mínimo esfuerzo (34). Se seleccionó como Gestor de Base de Datos, PostgreSQL en su versión 9.2. Como Entorno de Desarrollo Integrado se seleccionó JetBrains Pycharm 2018 ya que es el IDE que mejor se integra con el marco de trabajo seleccionado, permitiendo la ejecución de comandos y auto-completamiento de código. A continuación, se describen estas herramientas, tecnologías y lenguajes seleccionados.

1.5.1 Herramienta y lenguaje de modelado

El **Lenguaje Unificado de Modelado** (UML, del inglés *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y utilizado; está respaldado por el Object Management Group (35). UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados (36).

Se utiliza como lenguaje de modelado UML en su versión 2.0 para la confección del diagrama de componentes y del modelo de datos de la solución.

Se selecciona **Visual Paradigm for UML** en su versión 8.0 para la creación de los diagramas a modelar para la solución. Es una herramienta que puede integrarse con otras aplicaciones, como herramientas ofimáticas, permite generar código de forma automática reduciendo los tiempos de desarrollo y evitando errores en la codificación del software. Permite además generar diversos informes a partir de la información introducida en la herramienta (37).

1.5.2 Lenguaje de programación, marco de trabajo y entorno de desarrollo integrado

Python es un lenguaje de programación cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma. Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License, que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1 (34). Se selecciona para el desarrollo de la aplicación Python en su versión 3.6.

El marco de trabajo seleccionado para el desarrollo de la aplicación es **Django** en su versión 2.0.5, framework de desarrollo web de código abierto, escrito en Python, que respeta el patrón de diseño Modelo–Vista–Controlador. La meta fundamental de Django es facilitar la creación de sitios web complejos. Hace énfasis en el re-uso, la conectividad y extensibilidad de componentes y el desarrollo rápido. El lenguaje Python es usado en todas las partes del marco de trabajo, incluso en configuraciones, archivos, y en los modelos de datos.

Algunas características de Django son:

- Un mapeador objeto-relacional.
- Aplicaciones "enchufables" que pueden instalarse en cualquier página gestionada con Django.
- Una API de base de datos robusta.
- Un sistema incorporado de "vistas genéricas" que ahorra tener que escribir la lógica de ciertas tareas comunes.
- Un sistema extensible de plantillas basado en etiquetas, con herencia de plantillas.

- Un despachador de URLs basado en expresiones regulares.
- Un sistema "middleware" para desarrollar características adicionales; por ejemplo, la distribución principal de Django incluye componentes middleware que proporcionan cacheo, compresión de la salida, normalización de URLs, protección CSRF y soporte de sesiones.
- Soporte de internacionalización, incluyendo traducciones incorporadas de la interfaz de administración.
- Documentación incorporada accesible a través de la aplicación administrativa (incluyendo documentación generada automáticamente de los modelos y las bibliotecas de plantillas añadidas por las aplicaciones).

Un Entorno de Desarrollo Integrado, (IDE, del inglés *Integrated Development Environment*), proporciona servicios integrales para facilitarle al programador el desarrollo de software. Los IDE están diseñados para maximizar la productividad del programador proporcionando componentes muy unidos con interfaces de usuario similares. Presentan un único programa en el que se lleva a cabo todo el desarrollo. Generalmente, este programa suele ofrecer muchas características para la creación, modificación, compilación, implementación y depuración de software.

Se utiliza como IDE para el desarrollo de la aplicación **JetBrains Pycharm 2018.**

1.5.3 Sistema Gestor de Base de Datos (SGBD)

Un Sistema Gestor de Base de Datos (SGBD) es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos, además de proporcionar herramientas para añadir, eliminar, modificar y analizar los datos. Estos sistemas proporcionan métodos para mantener la integridad de los datos, administrar el acceso de los usuarios a estos y recuperar la información si fuese necesario (38).

PostgreSQL es un Sistema de Gestión de Bases de Datos relacional orientado a objetos y libre. Funciona bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema. Algunas de sus principales características son:

- Llaves ajenas o Claves Foráneas (foreign keys)
- Disparadores (triggers): Un disparador o trigger se define como una acción específica que se realiza de acuerdo a un evento, cuando éste ocurra dentro de la base de datos.

- Funciones: Bloques de código que se ejecutan en el servidor. Pueden ser escritos en varios lenguajes como: PL/PgSQ, C, C++, PL/Python, Java PL/Java web, entre otros.

Por las características antes mencionadas se selecciona como sistema gestor de base de datos PostgreSQL en su versión 9.2.

1.6 Patrones para el desarrollo de software

1.6.1 Patrones Arquitectónicos

Los patrones arquitectónicos, o patrones de arquitectura, también llamados arquetipos ofrecen soluciones a problemas de arquitectura de software en ingeniería de software. Dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. En comparación con los patrones de diseño, los patrones arquitectónicos tienen un nivel de abstracción mayor (39).

Para el desarrollo de la solución se selecciona el patrón Modelo–Plantilla–Vista (MTV), el cual separa los datos y la lógica de negocio de la interfaz de usuario, este patrón es una variación con la cual se trabaja en el marco de trabajo Django, del patrón Modelo Vista Controlador (40).

1.6.2 Patrones de Diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.

Un patrón de diseño resulta ser una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias (39).

1.6.2.1 Patrones Generales de Software para la Asignación de Responsabilidades (GRASP)

Los patrones GRASP (del inglés *General Responsibility Asignment Software Patterns*) describen los principios fundamentales de la asignación de responsabilidades a objetos (41). A continuación, se mencionan los patrones utilizados en el diseño de la solución:

- Experto: asigna una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.
- Creador**:** guía la asignación de responsabilidades relacionadas con la creación de objetos.
- Bajo acoplamiento: medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo acoplamiento no depende de muchas otras.
- Alta cohesión: medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme.
- Controlador: objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define el método de su operación.

1.6.3.1 Patrones GoF (Gang of Four)

Los patrones GoF pertenecen al campo del Diseño Orientado a Objetos. Están conformados por 23 patrones que se clasifican según su propósito en creacionales, estructurales y comportamiento (42).

- Creacionales: definen la forma en la que un objeto puede ser creado teniendo en cuenta la reutilización y mutabilidad. Estos describen la mejor manera de manejar instancias.
- Estructurales: describen cómo los objetos y las clases se pueden combinar para formar estructuras.
- Comportamiento: describen la forma de cómo organizar, administrar y combinar conductas y responsabilidades de objetos, centrándose en la comunicación entre ellos.

1.7 Verificación y validación de la herramienta

Para la evaluación de la herramienta se define una estrategia de prueba, donde se llevan a cabo los niveles de pruebas unitarias y pruebas de aceptación, así como los métodos de prueba de caja blanca y caja negra.

1.7.1 Pruebas unitarias

Las pruebas unitarias enfocan los esfuerzos de verificación en la unidad más pequeña del diseño de software. Las operaciones dentro de una clase son las unidades comprobables más pequeñas. Utiliza técnicas de prueba que ejercitan rutas específicas en una estructura de control para asegurar una cobertura completa y la máxima detección de errores (24).

1.7.2 Pruebas de aceptación

Las pruebas de aceptación son definidas por el cliente para cada historia de usuario. Se utilizan para validar que cada requisito implementado funciona como se había especificado (43).

1.7.3 Método de prueba de caja blanca

Las pruebas de caja blanca (conocidas como pruebas de caja de cristal o pruebas estructurales) se centran en los detalles procedimentales del software, por lo que su diseño está fuertemente ligado al código fuente. Es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba (24).

1.7.4 Método de prueba de caja negra

Las pruebas de caja negra se enfocan en los requisitos funcionales del software. Permiten derivar conjuntos de condiciones de entrada que revisarán por completo todos los requisitos funcionales para un programa (24).

Conclusiones parciales

Al finalizar el presente capítulo se arribó a las siguientes conclusiones parciales:

- Los análisis de factibilidad ayudan a identificar aquellos proyectos que tienen mayor probabilidad de éxito contribuyendo a una mejor identificación de riesgos en etapas tempranas del proyecto.
- El análisis de factibilidad en proyectos de software se puede considerar un problema de toma de decisiones que se desarrolla en un entorno de incertidumbre.
- Los criterios económicos, técnicos, comerciales y sociales son fundamentales en los análisis de factibilidad de proyectos de software.
- Los métodos de evaluación encontrados en sus mayorías son deterministas y no trabajan suficientemente la incertidumbre de la información, de ahí la necesidad de desarrollar una aplicación que permita evaluar proyectos informáticos teniendo en cuenta la factibilidad técnica, comercial y económica aplicando técnicas de softcomputing, pues estas contribuyen a modelar la imprecisión y a mejorar la capacidad de predicción de la factibilidad del proyecto de software.
- La metodología de software más adecuada para llevar a cabo el proceso de desarrollo es XP, dada la forma continua de interacción con el cliente, el poco tiempo de duración del proyecto y las características del equipo de desarrollo.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

Introducción al capítulo

En este capítulo se detalla la propuesta de solución y se establecen los roles que interactúan con el sistema. Se realiza el modelado del negocio y se muestran los requisitos funcionales identificados a través de entrevistas con el cliente en la fase de planificación. Se presentan como parte de la fase de planificación, las historias de usuario y el plan de iteraciones para el desarrollo del sistema. Se puntualizan la arquitectura y el diseño del software utilizando para ello las tarjetas Clase-Responsabilidad-Colaborador (CRC), el modelo de datos y los patrones de diseño aplicados.

2.1 Descripción de la propuesta de solución

La aplicación a desarrollar recibirá como entradas la información de un conjunto de proyectos a los que se le desea analizar la factibilidad, adicionados por el usuario con rol de Especialista. Recibirá, además, adicionados también por el rol mencionado anteriormente, el grupo de expertos que evaluarán para cada proyecto los criterios económicos, técnicos, sociales y comerciales seleccionados. La aplicación contendrá inicialmente un conjunto de criterios definidos por Peña (31), pero el Especialista podrá incluir otros a fin de adaptar el proceso de evaluación a cualquier proyecto. Los criterios tendrán un nivel de importancia definido por el Especialista. Los expertos podrán expresar sus evaluaciones en tres dominios: numérico, intervalar y lingüístico. Luego esas evaluaciones se unificarán en un dominio común a fin de facilitar el trabajo con ellas, utilizando las ecuaciones de transformación *1-10* formalizadas en el *epígrafe 1.3.2*, luego se transformarán a 2-tuplas utilizando las ecuaciones *11 y 12* y se agregarán las preferencias de todos los expertos para obtener el valor colectivo de cada criterio para cada proyecto evaluado utilizando el operador Media Aritmética Extendida, formalizado en la ecuación *13*. Para obtener la 2-tupla que representa el valor de factibilidad de cada proyecto y su precisión se agrega el valor de los criterios de cada proyecto considerando su peso y para ello se utiliza el operador Media Ponderada Extendida, formalizado en la ecuación *14*. Al final, si se evalúa más de un proyecto, se comparan las 2-tuplas obtenidas de cada uno de ellos y se muestra el listado ordenado de los mismos.

2.2 Roles del sistema

La **Tabla *2*** muestra los roles definidos para la interacción con el sistema según los diferentes niveles de acceso establecidos:

Tabla 2. Roles del sistema

Roles	Descripción
Administrador	Posee acceso pleno a todas las opciones de la aplicación.
Especialista	Registra en el sistema los proyectos a evaluar y el grupo de expertos que emitirán sus valoraciones sobre los criterios, asigna el valor de importancia a los criterios de evaluación y ejecuta el proceso de evaluación de factibilidad.
Experto	Evalúa por cada proyecto el conjunto de criterios definidos.

2.3 Modelado del negocio

Los procesos de negocio determinan la secuencia ordenada de actividades e información. La **Figura *3*** muestra el diagrama de proceso de negocio asociado al análisis de factibilidad de un proyecto de software.

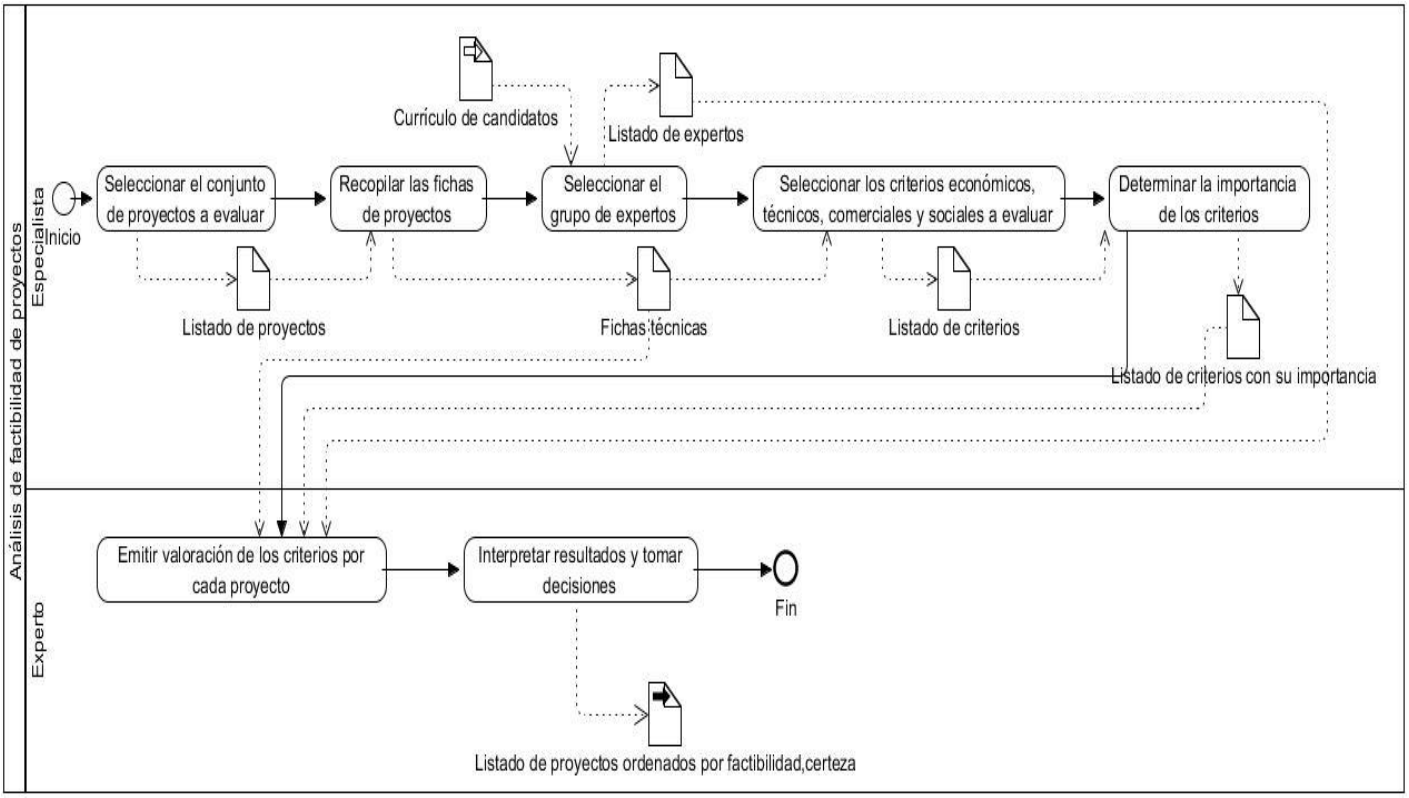


Figura 3. Diagrama de proceso del negocio.

El proceso de análisis de factibilidad comienza cuando el especialista de proyecto selecciona el conjunto de proyectos a evaluar y recopila las fichas asociadas a cada uno, obteniendo las fichas técnicas con las especificaciones técnicas, comerciales, económicas y sociales. Luego se selecciona el conjunto de expertos que participarán en la evaluación. Seguidamente se escogen los criterios económicos, técnicos, comerciales y sociales a evaluar determinando el valor de importancia para cada uno de ellos. A partir de las fichas técnicas y el listado de los criterios a evaluar con su importancia, los expertos emiten sus valoraciones de los criterios para cada proyecto a evaluar. Por último, con la evaluación obtenida para cada proyecto se analiza la información resultante y se toman las decisiones correspondientes de acuerdo con la naturaleza de los proyectos y las condiciones de la entidad desarrolladora.

2.4 Fase de planificación

El proceso de desarrollo de la herramienta de software surcará las fases propuestas por la metodología seleccionada, XP: planificación, diseño, desarrollo y pruebas.

En la fase de planificación se obtienen, a partir de entrevistas realizadas con el cliente, los requisitos que se desean se implementen como parte de la solución a partir del desarrollo de la herramienta informática. Las funcionalidades identificadas se describen de forma corta y sencilla en las historias de usuarios, y se evalúa el tiempo de desarrollo de cada una de estas por parte de los desarrolladores. Luego, mediante un plan de iteraciones es posible planificar el tiempo estimado de duración del proceso de desarrollo de la herramienta.

2.4.1 Requisitos del software

Como parte del análisis previo al diseño de la solución se emplearon las siguientes técnicas de obtención o captura de requisitos:

La **entrevista** es una de las técnicas más aceptadas dentro de la ingeniería de requisitos. Permite tomar conocimiento del problema y comprender los objetivos de la solución buscada, para tener una amplia visión de las necesidades del usuario (44). Se realizaron entrevistas con el cliente para conocer el negocio y obtener la información necesaria acerca de las necesidades y perspectivas para desarrollar la solución. Finalmente se acordó la implementación de una herramienta de apoyo a la toma de decisiones de los principales especialistas de los centros productivos, y facilitar con ella el análisis de estudios de factibilidad de proyectos de software.

2.4.1.1 Requisitos funcionales

Los requisitos funcionales de un sistema refieren lo que el sistema debe hacer. Se describen generalmente de forma abstracta que entiendan los usuarios del sistema. Detallan las funciones del sistema, sus entradas, salidas y sus excepciones (45).

En entrevistas realizadas al cliente se identificaron un total de 44 requisitos funcionales (RF). La prioridad de cada requisito fue definida por el cliente en función de la importancia para el negocio. A continuación, se listan los requisitos funcionales:

Tabla 3.Requisitos funcionales

Número	Requisitos Funcionales	Prioridad
RF-1	Adicionar usuario	Media
RF-2	Modificar usuario	Media
RF-3	Eliminar usuario	Media
RF-4	Listar usuarios	Baja
RF-5	Buscar usuario	Baja
RF-6	Adicionar rol	Media
RF-7	Modificar rol	Media
RF-8	Eliminar rol	Media
RF-9	Listar roles	Baja
RF-10	Buscar rol	Baja
RF-11	Adicionar ficha técnica del proyecto	Alta
RF-12	Modificar ficha técnica del proyecto	Alta
RF-13	Eliminar ficha técnica del proyecto	Media
RF-14	Mostrar ficha técnica del proyecto	Media
RF-15	Listar fichas técnicas de proyectos	Baja
RF-16	Buscar ficha técnica de un proyecto	Baja
RF-17	Adicionar criterios técnicos de una ficha de proyecto	Alta

RF-18	Modificar criterios técnicos de una ficha de proyecto	Alta
RF-19	Mostrar criterios técnicos de una ficha de proyecto	Media
RF-20	Adicionar criterios comerciales de una ficha de proyecto	Alta
RF-21	Modificar criterios comerciales de una ficha de proyecto	Alta
RF-22	Mostrar criterios comerciales de una ficha de proyecto	Media
RF-23	Adicionar flujos de caja de una ficha de proyecto	Alta
RF-24	Modificar flujos de caja de una ficha de proyecto	Alta
RF-25	Mostrar flujos de caja de una ficha de proyecto	Media
RF-26	Adicionar criterios sociales de una ficha de proyecto	Alta
RF-27	Modificar criterios sociales de una ficha de proyecto	Alta
RF-28	Mostrar criterios sociales de una ficha de proyecto	Media
RF-29	Calcular criterios económicos VAN, TIR y PRI	Alta
RF-30	Crear evaluación de proyectos	Alta
RF-31	Modificar evaluación de proyectos	Alta
RF-32	Eliminar evaluación de proyectos	Media
RF-33	Mostrar resultado de la evaluación de proyectos	Alta
RF-34	Listar evaluaciones de proyectos	Baja
RF-35	Buscar evaluación de proyectos	Baja
RF-36	Asociar proyecto a una evaluación	Alta
RF-37	Eliminar proyecto de una evaluación	Media
RF-38	Asociar experto a una evaluación	Alta
RF-39	Eliminar experto de una evaluación	Media
RF-40	Asociar criterios a evaluar	Alta
RF-41	Eliminar criterios de una evaluación	Media
RF-42	Añadir valor de importancia de los criterios	Alta

RF-43	Evaluar criterios	Alta
RF-44	Generar listado ordenado de proyectos evaluados	Alta

2.4.1.2 Requisitos no funcionales

Los requisitos no funcionales se relacionan directamente con los servicios específicos que el sistema entrega a los usuarios. Se relacionan con propiedades procedentes del sistema como fiabilidad, tiempo de respuesta y uso de almacenamiento. Pueden definir restricciones sobre la implementación del sistema, como las capacidades de los dispositivos. Provienen de características requeridas del software, la organización que desarrolla el software o de fuentes externas *(45)*. A continuación, se muestran los requisitos no funcionales de la herramienta según la clasificación propuesta por Sommerville:

Usabilidad

RNF-1. La herramienta informática a desarrollar será una aplicación web.

RNF-2. La herramienta debe presentar una vista sencilla, descriptiva y fácil de usar, con el objetivo de proporcionar a los usuarios un mejor entendimiento con la aplicación.

Disponibilidad

RNF-3. La herramienta deberá estar disponible siempre, asegurando el acceso desde cualquier lugar de red a los usuarios autorizados.

Portabilidad

RNF-4. La herramienta debe ser multiplataforma para poder ser instalada en los sistemas operativos Windows y Linux/Unix.

Confiabilidad

RNF-5. La herramienta restringirá el acceso por roles a los usuarios autorizados.

Eficiencia

RNF-6. El tiempo de respuesta en el procesamiento de los datos y solicitud de estos no debe sobrepasar los 5 segundos.

Seguridad

RNF-7. Existirán diferentes tipos de usuarios según las acciones que puedan realizar. Los permisos serán limitados, basados en los roles definidos y el usuario no podrá gestionarlos.
RNF-8. La herramienta debe permitir que la contraseña se almacene de manera encriptada en la base de datos.
RNF-9. La herramienta debe permitir la comprobación de credenciales en la autenticación del usuario en el servidor de aplicaciones y no en el servidor de base de datos, para evitar inyecciones SQL que falseen la autenticación y provoquen la suplantación de identidad.
RNF-10. El sistema debe proveer protección contra inyecciones SQL.
RNF-11. El sistema debe proveer protección contra ataques XSS.

Desarrollo

RNF-12. La herramienta será desarrollada con el lenguaje de programación Python, utilizando el framework web Django.
RNF-13. La herramienta utilizará como gestor de base de datos PostgreSQL 9.4.
RNF-14. La herramienta utilizará para el desarrollo de las interfaces HTML5, CSS3 y JavaScript.

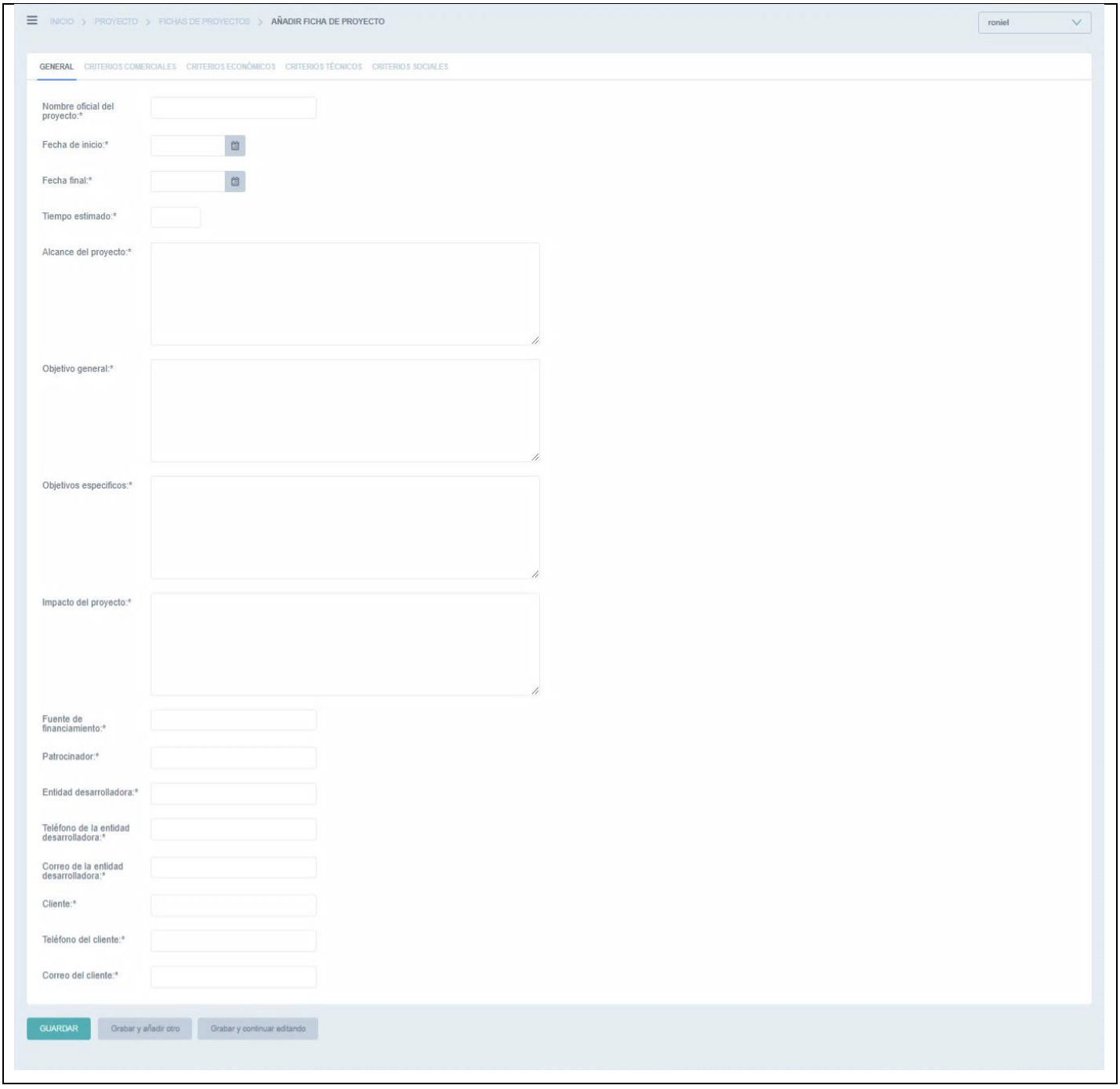
2.4.2 Historias de usuario

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. Se describe brevemente las características que el sistema debe poseer en cuanto a los requisitos funcionales. Cada historia de usuario es lo adecuadamente comprensible y delimitada para que los programadores puedan implementarla en unas semanas (46). A continuación, se muestra la historia de usuario correspondiente al requisito funcional *Adicionar ficha técnica del proyecto*.

Tabla 4. Historia de usuario *Adicionar ficha técnica del proyecto.*

HISTORIA DE USUARIO	
Número: HU11	**Nombre historia de usuario:** Adicionar ficha técnica del proyecto
Usuario: Roniel López Álvarez	**Iteración asignada:** 1
Prioridad en negocio: Alta	**Puntos estimados:** 2 días
Riesgo en desarrollo:	**Puntos reales:** 2 días
Descripción:	

El especialista registra la ficha técnica con la información del proyecto a evaluar en el sistema. El sistema muestra una interfaz para insertar la ficha técnica del proyecto.
Observaciones: La funcionalidad correspondiente al requisito funcional solo puede ser realizada por los roles especialista y administrador.
Interfaz de usuario:


INICIO > PROYECTO > FICHAS DE PROYECTOS > AÑADIR FICHA DE PROYECTO
roniel
GENERAL
CRITERIOS COMERCIALES
CRITERIOS ECONÓMICOS
CRITERIOS TÉCNICOS
CRITERIOS SOCIALES
Nombre oficial del proyecto:*
Fecha de inicio:*
Fecha final:*
Tiempo estimado:*
Alcance del proyecto:*
Objetivo general:*
Objetivos específicos:*
Impacto del proyecto:*
Fuente de financiamiento:*
Patrocinador:*
Entidad desarrolladora:*
Teléfono de la entidad desarrolladora:*
Correo de la entidad desarrolladora:*
Cliente:*
Teléfono del cliente:*
Correo del cliente:*
GUARDAR
Grabar y añadir otro
Grabar y continuar editando

2.4.3 Plan de iteraciones

Se desarrollan y prueban en un ciclo de iteración un conjunto de historias de usuarios seleccionadas de acuerdo al orden preestablecido. Cada historia de usuario se convierte en una tarea específica de programación (43). Para la implementación del sistema se dividieron las historias de usuario en cuatro iteraciones, teniendo en cuenta la prioridad. En una primera y segunda iteración se desarrollaron las historias de usuario de prioridad alta para el negocio, en una tercera las de prioridad media y finalmente en una cuarta iteración las de prioridad baja. Se muestra en la **Tabla *5*** el plan de iteraciones definido.

Tabla 5. Plan de iteraciones.

Iteración	Historia de usuario	Prioridad	Duración de HU (días)	Duración total (semanas)
1	Adicionar ficha de proyecto	Alta	3	2
	Adicionar evaluación de proyectos	Alta	3	
	Mostrar resultado de la evaluación de proyectos	Alta	2	
	Agregar proyecto a una evaluación	Alta	1	
	Agregar experto a una evaluación	Alta	1	
	Agregar criterio comercial a evaluar	Alta	1	
	Agregar criterio técnico a evaluar	Alta	1	
	Agregar criterio social a evaluar	Alta	1	
	Adicionar flujos de caja de una ficha de proyecto	Alta	1	
	Calcular criterios económicos VAN, TIR y PRI	Alta	1	
2	Añadir valor de importancia de los criterios	Alta	3	2
	Modificar criterios técnicos de una ficha de proyecto	Alta	1	
	Modificar criterios comerciales de una ficha de proyecto	Alta	1	
	Modificar flujos de caja de una ficha de proyecto	Alta	1	
	Modificar criterios sociales de una ficha de proyecto		1	

	Añadir valor de importancia de los criterios	Alta	1	
	Evaluar criterio por parte del experto	Alta	2	
	Aplicar modelo 2-tuplas para la evaluación de criterios técnicos y comerciales	Alta	3	
	Generar listado ordenado de proyectos evaluados	Alta	2	
3	Adicionar usuario	Media	2	5
	Modificar usuario	Media	2	
	Eliminar usuario	Media	1	
	Adicionar rol	Media	1	
	Modificar rol	Media	1	
	Eliminar rol	Media	1	
	Modificar ficha de proyecto	Media	2	
	Eliminar ficha de proyecto	Media	1	
	Mostrar ficha de proyecto	Media	1	
	Mostrar criterios técnicos de una ficha de proyecto	Media	2	
	Mostrar criterios comerciales de una ficha de proyecto	Media	2	
	Mostrar flujos de caja de una ficha de proyecto	Media	2	
	Mostrar criterios sociales de una ficha de proyecto	Media	1	
	Eliminar evaluación de proyectos	Media	1	
	Eliminar proyecto de una evaluación	Media	1	
	Eliminar experto de una evaluación	Media	1	
	Eliminar criterio comercial de una evaluación	Media	1	
	Eliminar criterio técnico de una evaluación	Media	1	
	Exportar resultado de la evaluación	Media	3	
	Listar usuarios	Baja	1	

4	Buscar usuario	Baja	1	2
	Listar roles	Baja	1	
	Buscar rol	Baja	1	
	Listar fichas de proyectos	Baja	1	
	Buscar ficha de proyecto	Baja	1	
	Listar evaluaciones de proyectos	Baja	1	
	Buscar evaluación de proyectos	Baja	1	
Total				11

2.5 Fase de diseño

En la fase de diseño se realiza la confección de las tarjetas Clase Responsabilidad Colaborador (CRC). Se selecciona la arquitectura para el desarrollo de la herramienta.

2.5.1 Tarjetas Clase Responsabilidad Colaborador (CRC)

Las tarjetas CRC contienen la información del nombre de la clase, sus responsabilidades y sus colaboradores. Las responsabilidades de una clase están dadas por los elementos que conoce, sus atributos y métodos. Los colaboradores de una clase son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades. La **Tabla *6*** muestra la tarjeta CRC correspondiente a la clase Proyecto.

Tabla 6. Tarjeta CRC de la clase Proyecto.

TARJETA CRC	
Clase: Proyecto	
Descripción: Contiene la información de los proyectos a evaluar	
Atributos	
Nombre	**Descripción**
id	Campo identificador
nombre_oficial	Nombre oficial del proyecto
fecha_inicio	Fecha de inicio del proyecto
fecha_fin	Fecha final planificada

tiempo_estimado	Tiempo estimado de duración del proyecto
alcance_del_proyecto	Alcance del proyecto
objetivo_general	Objetivo general del proyecto
objetivos_especificos	Objetivos específicos del proyecto
impacto_del_proyecto	Impacto social del proyecto
fuente_de_financiamiento	Fuente de financiamiento del proyecto
patrocinador	Patrocinador del proyecto
entidad_desarrolladora	Entidad que desarrolla el proyecto
telefono_entidad_desarrolladora	Teléfono de la entidad que desarrolla el proyecto
correo_entidad_desarrolladora	Correo de la entidad que desarrolla el proyecto
cliente	Cliente que solicita el proyecto
telefono_cliente	Teléfono del cliente que solicita el proyecto
correo_cliente	Correo del cliente que solicita el proyecto
Responsabilidades:	**Colaboraciones:**
Contener la información de los proyectos a evaluar	FichaTecnica, FichaComercial, FichaSocial, FichaEconomica

2.5.2 Modelo de datos

El modelo de datos está basado en una percepción del mundo real consistente en objetos básicos llamados entidades y de relaciones entre estos objetos. Es uno de los diferentes modelos de datos semánticos, pues representa el significado de los datos (47). La ***Figura 4*** muestra el modelo de datos correspondiente a la solución compuesto por 14 tablas, representando los datos, atributos y relaciones entre ellos.

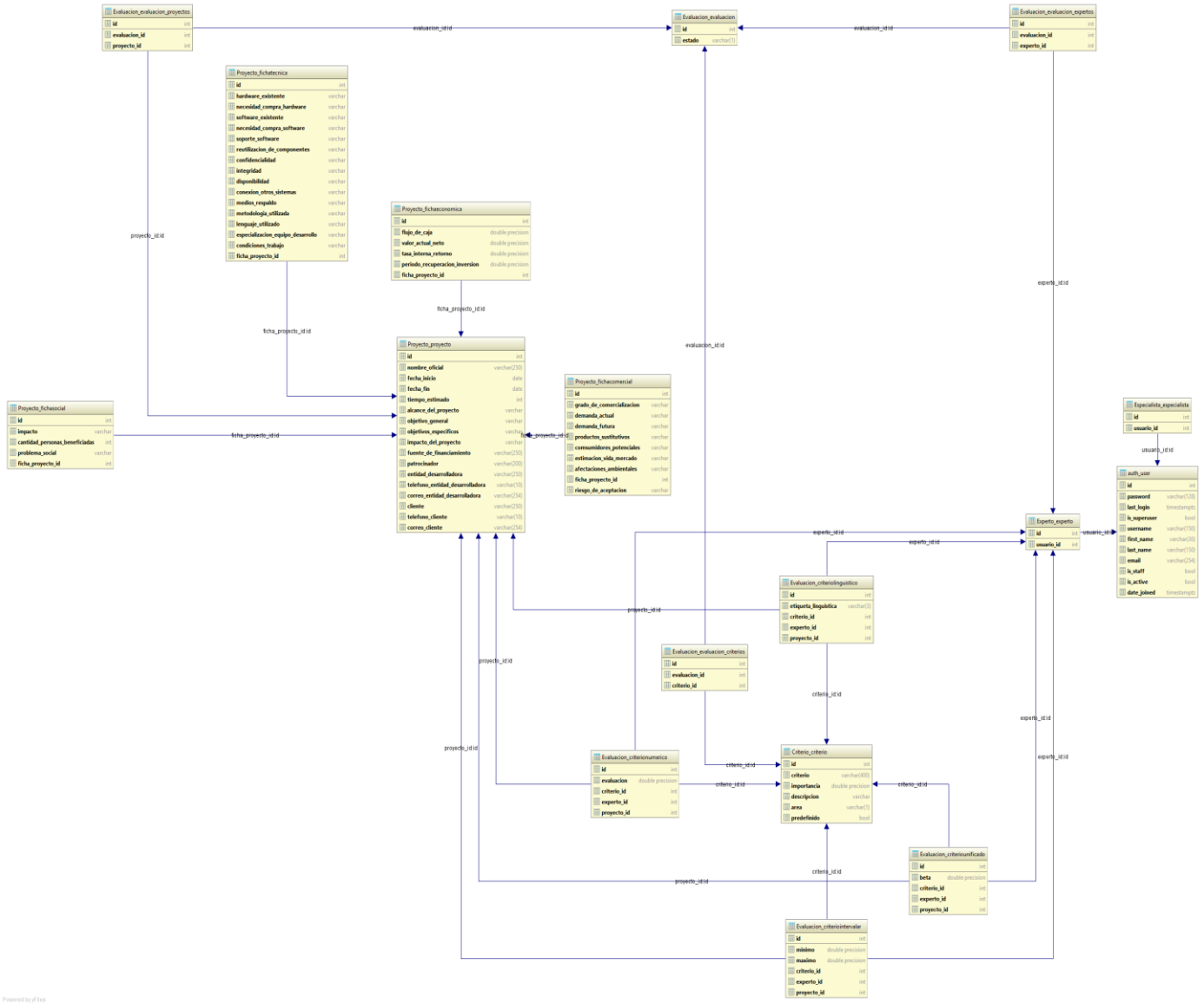


Figura 4. Modelo de datos de la solución.

2.5.3 Arquitectura del sistema

La arquitectura definida para la herramienta es Modelo Plantilla Vista (MTV), patrón arquitectónico en el cual se basa Django. Este patrón organiza el código separando los datos y la lógica del negocio de la interfaz de usuario en tres componentes: el modelo, la plantilla y el controlador.

Modelo (Model): capa de acceso a la base de datos. Contiene la información sobre los datos, cómo acceder a estos y las relaciones entre ellos.
Plantilla (Template): capa de presentación. Contiene las decisiones relacionadas a la presentación: como se muestra la información en una página web.
Vista (View): capa de la lógica de negocios. Contiene la lógica que accede al modelo y la delega a la plantilla apropiada.

La **Figura *5*** muestra la representación de la estructura y composición de la herramienta mediante el patrón arquitectónico seleccionado, a través de un diagrama de componentes.

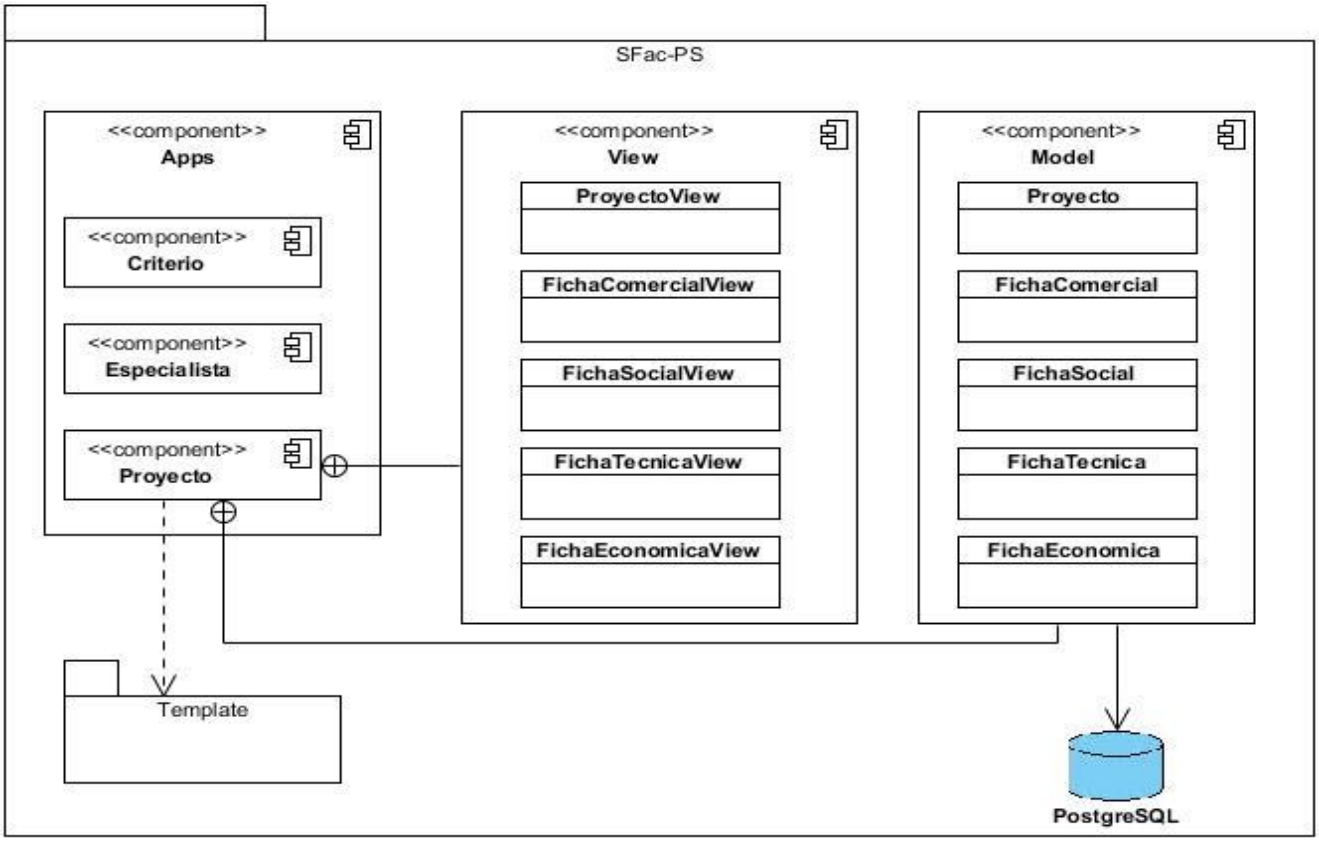


Figura 5. Arquitectura de la herramienta mediante un diagrama de paquetes.

2.5.4 Patrones de diseño

A continuación, se describen los patrones GRASP y GoF empleados en el diseño de las clases de la herramienta:

2.5.4.1 Patrones Generales de Software para la Asignación de Responsabilidades (GRASP)

Los patrones GRASP empleados en el desarrollo de la solución son:

Controlador: se ve presente en las vistas de la herramienta, las cuales son las responsables de implementar las funcionalidades correspondientes a los requisitos funcionales. En la **Figura *6*** se muestra el uso del patrón en las vistas de la app *Evaluación*.

```
class CriterioNumericoCreate(CreateView):...
class CriterioNumericoDetail(DetailView):...
class CriterioNumericoDelete(DeleteView):...
class CriterioNumericoUpdate(UpdateView):...
class CriterioNumericoList(ListView):...
class CriterioIntervalarCreate(CreateView):...
class CriterioIntervalarDetail(DetailView):...
class CriterioIntervalarUpdate(UpdateView):...
class CriterioIntervalarDelete(DeleteView):...
class CriterioIntervalarList(ListView):...
class CriterioLinguisticoCreate(CreateView):...
class CriterioLinguisticoDetail(DetailView):...
class CriterioLinguisticoUpdate(UpdateView):...
```

Figura 6. Patrón Controlador en las vistas de la app Evaluación.

Creador: se pone de manifiesto en las vistas de la herramienta. Ofrece soporte a un bajo acoplamiento y una alta cohesión, contribuyendo a la reutilización de las clases. La **Figura *7*** muestra el uso del patrón mediante la vista *evaluarCriterio*.

```
def evaluarCriterio(request, dominio="N"):
    if (request.method == 'POST' and dominio=="N"):
        form = EvaluarCriterioNumericoForm(request.POST)
        if form.is_valid():
            c = form.save(commit=False)
            c.experto = Experto.objects.get(usuario=request.user.id)
            c.save()
            form.save_m2m()
            return redirect(to='/')
```

Figura 7. Patrón Creador en la vista evaluarCriterio.

Experto: las clases entidades contienen la información relacionada con los objetos persistentes que representan, por lo que este patrón resulta de gran utilidad en las clases del modelo. La

```
class Proyecto(models.Model):
    nombre_oficial = models.CharField(max_length=250, verbose_name='Nombre oficial del proyecto')
    fecha_inicio = models.DateField(verbose_name='Fecha de inicio')
    fecha_fin = models.DateField(verbose_name='Fecha final')
    tiempo_estimado = models.PositiveIntegerField()
    alcance_del_proyecto = models.TextField()
    objetivo_general = models.TextField()
    objetivos_especificos = models.TextField()
    impacto_del_proyecto = models.TextField()
    fuente_de_financiamiento = models.CharField(max_length=250)
    patrocinador = models.CharField(max_length=200)
    entidad_desarrolladora = models.CharField(max_length=250)
    telefono_entidad_desarrolladora = models.CharField(max_length=10, verbose_name='Teléfono de la entidad '
                                                                                    'desarrolladora')
    correo_entidad_desarrolladora = models.EmailField(verbose_name='Correo de la entidad desarrolladora')
    cliente = models.CharField(max_length=250)
    telefono_cliente = models.CharField(max_length=10, verbose_name='Teléfono del cliente')
    correo_cliente = models.EmailField(verbose_name='Correo del cliente')

    def __str__(self):
        return self.nombre_oficial

    class Meta:
        verbose_name = "Ficha de Proyecto"
        verbose_name_plural = "Fichas de Proyectos"
```

***Figura* 8** muestra el empleo del patrón en la clase entidad Proyecto.

```
class Proyecto(models.Model):
    nombre_oficial = models.CharField(max_length=250, verbose_name='Nombre oficial del proyecto')
    fecha_inicio = models.DateField(verbose_name='Fecha de inicio')
    fecha_fin = models.DateField(verbose_name='Fecha final')
    tiempo_estimado = models.PositiveIntegerField()
    alcance_del_proyecto = models.TextField()
    objetivo_general = models.TextField()
    objetivos_especificos = models.TextField()
    impacto_del_proyecto = models.TextField()
    fuente_de_financiamiento = models.CharField(max_length=250)
    patrocinador = models.CharField(max_length=200)
    entidad_desarrolladora = models.CharField(max_length=250)
    telefono_entidad_desarrolladora = models.CharField(max_length=10, verbose_name='Teléfono de la entidad '
                                                                                    'desarrolladora')
    correo_entidad_desarrolladora = models.EmailField(verbose_name='Correo de la entidad desarrolladora')
    cliente = models.CharField(max_length=250)
    telefono_cliente = models.CharField(max_length=10, verbose_name='Teléfono del cliente')
    correo_cliente = models.EmailField(verbose_name='Correo del cliente')

    def __str__(self):
        return self.nombre_oficial

    class Meta:
        verbose_name = "Ficha de Proyecto"
        verbose_name_plural = "Fichas de Proyectos"
```

Figura 8. Patrón Experto en la clase Proyecto.

Alta cohesión: las clases de la herramienta tienen sus responsabilidades relacionadas entre sí, no efectúan un trabajo excesivo, lo que permite reducir el mantenimiento de la herramienta y aumentar la reutilización. El objetivo fundamental de la filosofía de diseño de Django es el bajo acoplamiento y la alta cohesión.

Bajo acoplamiento: cada clase de la herramienta tiene el menor número de dependencias posibles, una de ellas solamente solicita a otra en caso de que exista referencia dentro de sus atributos, lo que permite robustez en la herramienta. Por ejemplo, el sistema de plantillas no se relaciona con las solicitudes web, la visualización de datos es independiente de la capa de la base de datos y las vistas funcionan sin tener en cuenta qué sistema de plantilla utiliza un programador.

2.5.4.2 Patrones GoF

Los patrones GoF empleados en el desarrollo de la solución son:

Estructurales

Decorador (del inglés Decorator)*:** aplicado a la generación de interfaces, la solución que ofrece este patrón es la de añadir funcionalidad adicional a las plantillas. Ejemplo, añadir partes de código contenidos en otros ficheros; se trata de decorar las plantillas con elementos adicionales reutilizables. El sistema de plantillas de Django está provisto de un mecanismo de herencia gracias al cual la decoración de plantillas resulta de una flexibilidad y versatilidad total. En la **Figura *9 se muestra el uso del patrón.

```
{% extends "admin/index.html" %}
{% block content %}
    <div id="content" class="colM">
        <div class="dashboard-container columns_3 cf">
            {% include 'Especialista/portal/portal_c1.html' %}
            {% include 'Especialista/portal/portal_c2.html' %}
            {% include 'Especialista/portal/portal_c3.html' %}
        </div>
    </div>
{% endblock %}
{% block sidebar %}
    {{ block.super }}
{% endblock %}
```

Figura 9. Uso del patrón Decorador en la generación de interfaces.

Este patrón aplicado a las vistas añade una funcionalidad extra a las mismas, sin necesidad de reescribir código. Ejemplo de ello se muestra en la **Figura *10***.

```
@require_http_methods(['GET','POST'])
def evaluarCriterio(request, dominio="N"):
```

Figura 10. Uso del patrón Decorador en las vistas.

Creacionales

Método de Fábrica (del inglés Factory Method)*:** define una interfaz para la creación de un objeto, lo que permite a las subclases decidir de qué clase instanciarlo. Permite que una clase difiera la instanciación en favor de sus subclases. En la herramienta se manifiesta el uso del patrón en la clase *CrearFormularioEvaluacion*, la cual decide el tipo de formulario a crear, así como los tipos de datos correspondientes a los campos del formulario. La **Figura *11 muestra el empleo del patrón.

```
class CrearFormularioEvaluacion:

    def __init__(self, dominio):
        self.dominio = dominio

    def getForm(self):
        if self.dominio == 'N':
            return EvaluarCriterioNumericoForm()
        elif self.dominio == 'I':
            return  EvaluarCriterioIntervalarForm()
        else:
            return EvaluarCriterioLinguisticoForm()
```

Figura 11. Uso del patrón Método de Fábrica.

Comportamiento

Mediador (del inglés Mediator): Las vistas son mediadoras entre los modelos y las plantillas.

Mediante el uso de los patrones de diseño se garantizó la asignación de las responsabilidades correspondientes a cada una de las clases, y el menor número de dependencias y relaciones entre estas, lo que contribuye a un correcto diseño del sistema, y facilita su implementación y reutilización.

Conclusiones parciales

Al concluir el capítulo se llegó a las siguientes conclusiones:

- El diagrama de procesos de negocio contribuye al entendimiento del proceso de análisis de factibilidad de proyectos de software.
- Los artefactos generados en las fases de planificación y diseño favorecen el desarrollo exitoso de la herramienta informática a implementar para la toma de decisiones en los análisis de factibilidad.
- El modelo de representación lingüística basado en 2-tuplas para la evaluación de criterios técnicos, comerciales, sociales y económicos, brinda una solución favorable para la unificación y tratamiento de la información heterogénea de los resultados sin pérdida de información en el proceso.

CAPÍTULO 3: DESARROLLO Y PRUEBAS

Introducción al capítulo

En este capítulo se evidencian los artefactos generados en la fase de desarrollo de la metodología de software seleccionada, como son las tareas de ingeniería. Se describen los estándares de codificación utilizados durante la implementación del sistema para garantizar un buen entendimiento y claridad del código. Por último, se describen las pruebas realizadas a la herramienta.

3.1 Fase de Desarrollo

En la fase de desarrollo se implementa el código de la herramienta diseñada, como resultado final se obtiene la herramienta correspondiente a la solución del problema identificado. En esta fase se realiza la planificación y ejecución de las tareas de ingeniería.

3.1.1 Tareas de ingeniería por historia de usuario

Las tareas de ingeniería son actividades asociadas a la implementación de las historias de usuario. A continuación en **Tabla *7***, se muestran las tareas a desarrollar para la implementación, por cada historia de usuario.

Tabla 7. Tareas de ingeniería por historia de usuario

Iteración	Historia de usuario	Tareas de ingeniería
1	Adicionar ficha de proyecto	Implementar una funcionalidad que permita a un especialista adicionar una ficha técnica de un proyecto en el sistema
	Adicionar evaluación de proyectos	Implementar una funcionalidad que permita a un especialista comenzar un proceso de evaluación de proyectos en el sistema
	Mostrar resultado de la evaluación de proyectos	Implementar una funcionalidad que permita mostrar el resultado de la evaluación realizada por un especialista, una vez evaluados los criterios por parte de los expertos
	Agregar proyecto a una evaluación	Implementar una funcionalidad que permita a un especialista agregar un proyecto a una evaluación

	Agregar experto a una evaluación	Implementar una funcionalidad que permita a un especialista agregar los expertos a una evaluación
	Agregar criterio comercial a evaluar	Implementar una funcionalidad que permita a un especialista agregar los criterios comerciales a tener en cuenta para la evaluación de los proyectos
	Agregar criterio técnico a evaluar	Implementar una funcionalidad que permita a un especialista agregar los criterios técnicos a tener en cuenta para la evaluación de los proyectos
	Agregar criterio social a evaluar	Implementar una funcionalidad que permita a un especialista agregar los criterios sociales a tener en cuenta para la evaluación de los proyectos
	Adicionar flujos de caja de una ficha de proyecto	Implementar una funcionalidad que permita a un especialista agregar flujos de caja a tener en cuenta para la evaluación de los proyectos
	Calcular criterios económicos VAN, TIR y PRI	Implementar una funcionalidad que permita calcular los criterios económicos VAN, TIR y PRI de los proyectos
2	Añadir valor de importancia de los criterios	Implementar una funcionalidad que permita a un especialista añadir los valores de importancia de los criterios seleccionados para la evaluación
	Modificar criterios técnicos de una ficha de proyecto	Implementar una funcionalidad que permita a un especialista modificar los criterios técnicos de una ficha de proyecto
	Modificar criterios comerciales de una ficha de proyecto	Implementar una funcionalidad que permita a un especialista modificar los criterios comerciales de una ficha de proyecto
	Modificar flujos de caja de una ficha de proyecto	Implementar una funcionalidad que permita a un especialista modificar los flujos de caja de una ficha de proyecto
	Modificar criterios sociales de una ficha de proyecto	Implementar una funcionalidad que permita a un especialista modificar los criterios sociales de una ficha de proyecto
	Evaluar criterio por parte del experto	Implementar una funcionalidad que permita a un experto emitir una evaluación para cada criterio seleccionado

	Aplicar modelo 2-tuplas para la evaluación de criterios	Implementar en Python los operadores de agregación del modelo lingüístico 2-tuplas
	Generar listado ordenado de proyectos evaluados	Implementar una funcionalidad para generar el listado de los proyectos evaluados
3	Adicionar usuario	Implementar una funcionalidad que permita a un administrador adicionar un usuario en el sistema
	Modificar usuario	Implementar una funcionalidad que permita a un administrador modificar un usuario adicionado en el sistema
	Eliminar usuario	Implementar una funcionalidad que permita a un administrador eliminar un usuario del sistema
	Adicionar rol	Implementar una funcionalidad que permita a un administrador adicionar un rol en el sistema
	Modificar rol	Implementar una funcionalidad que permita a un administrador modificar un rol adicionado en el sistema
	Eliminar rol	Implementar una funcionalidad que permita a un administrador eliminar un rol del sistema
	Modificar ficha de proyecto	Implementar una funcionalidad que permita a un especialista modificar una ficha de proyecto del sistema
	Eliminar ficha de proyecto	Implementar una funcionalidad que permita a un especialista eliminar una ficha de proyecto del sistema
	Mostrar ficha de proyecto	Implementar una funcionalidad que permita mostrar los datos de una ficha de proyecto adicionada en el sistema
	Mostrar criterios técnicos de una ficha de proyecto	Implementar una funcionalidad que permita mostrar los criterios técnicos de una ficha de proyecto adicionada en el sistema
	Mostrar criterios comerciales de una ficha de proyecto	Implementar una funcionalidad que permita mostrar los criterios comerciales de una ficha de proyecto adicionada en el sistema
	Mostrar criterios sociales de una ficha de proyecto	Implementar una funcionalidad que permita mostrar los criterios sociales de una ficha de proyecto adicionada en el sistema

	Mostrar criterios económicos de una ficha de proyecto	Implementar una funcionalidad que permita mostrar los criterios económicos de una ficha de proyecto adicionada en el sistema
	Eliminar evaluación de proyectos	Implementar una funcionalidad que permita eliminar una evaluación del sistema
	Eliminar proyecto de una evaluación	Implementar una funcionalidad que permita eliminar un proyecto de una evaluación del sistema
	Eliminar experto de una evaluación	Implementar una funcionalidad que permita eliminar un experto de una evaluación del sistema
	Eliminar criterios de una evaluación	Implementar una funcionalidad que permita eliminar criterios de una evaluación del sistema
4	Listar usuarios	Implementar una funcionalidad que permita listar los usuarios adicionados en el sistema.
	Buscar usuario	Implementar una funcionalidad que permita buscar un usuario adicionado en el sistema.
	Listar roles	Implementar una funcionalidad que permita listar los roles adicionados en el sistema.
	Buscar rol	Implementar una funcionalidad que permita buscar un rol adicionado en el sistema.
	Listar fichas de proyectos	Implementar una funcionalidad que permita listar las fichas de proyectos adicionadas en el sistema.
	Buscar ficha de proyecto	Implementar una funcionalidad que permita buscar una ficha de proyecto adicionada en el sistema.
	Listar evaluaciones de proyectos	Implementar una funcionalidad que permita listar las evaluaciones adicionadas en el sistema.
	Buscar evaluación de proyectos	Implementar una funcionalidad que permita buscar una evaluación adicionada en el sistema.

3.2.1.1 Tareas de ingeniería

En las **Tabla *8*** y **Tabla *9*** se muestran las tareas de ingeniería correspondientes a las historias de usuario *Adicionar ficha técnica del proyecto* y *Crear evaluación de proyectos*.

Tabla 8. Tarea de ingeniería detallada 1.

TAREA DE INGENIERÍA	
Número tarea: 1	**Historia de usuario:** Adicionar ficha técnica del proyecto
Nombre tarea: Implementar funcionalidad Adicionar ficha técnica de un proyecto	
Tipo de tarea: Desarrollo **(Desarrollo, Corrección, Mejora)**	**Puntos estimados (días):** 2
Fecha inicio: 05/03/2015	**Fecha fin:** 07/03/2015
Programador responsable: Roniel López Álvarez	
Descripción: Implementar la funcionalidad correspondiente al requisito funcional adicionar ficha técnica del proyecto. La acción correspondiente será realizada por el especialista del proyecto.	

Tabla 9. Tarea de ingeniería detallada 12

TAREA DE INGENIERÍA	
Número tarea: 12	**Historia de usuario:** Crear evaluación de proyectos
Nombre tarea: Implementar funcionalidad Crear evaluación de proyectos	
Tipo de Tarea: Desarrollo **(Desarrollo, Corrección, Mejora)**	**Puntos estimados (días):** 2
Fecha inicio: 12/03/2015	**Fecha fin:** 13/03/2015
Programador responsable: Roniel López Álvarez	
Descripción: Implementar la funcionalidad correspondiente a la creación de una evaluación por parte del especialista.	

3.1.2 Estándares de codificación

En función de lograr estandarización en el código del sistema, se definen un conjunto de pautas a seguir durante la implementación.

Definición de clases

Se utiliza el estilo de codificación "*UpperCamelCase"* para nombrar las clases, el cual establece que los nombres inician con letra mayúscula y si poseen más de una palabra, la primera letra de estas deberá ser mayúsculas también. El uso de este estilo se evidencia en la **Figura *12***.

```
class Proyecto(models.Model):...

class FichaComercial(models.Model):...

class FichaSocial(models.Model):...

class FichaTecnica(models.Model):...

class FichaEconomica(models.Model):...
```

Figura 12. Definición de clases.

Definición de variables y métodos

Los nombres de las variables deben ser descriptivos y concisos. No se usan abreviaciones. Se utiliza el estilo de codificación “lowerCamelCase”, el cual establece que los nombres inician con letra minúscula y cada nueva palabra debe iniciar con mayúscula. El uso de este estilo se evidencia en la **Figura *13***.

```
def unificarCriterios(request, evaluacion_id):...

def agregacion(request, valores):...

def ordenarResultados(request, v_proyectos):...

def resultados(request):...

def resultadosDetail(request, pk):
    return unificarCriterios(request, pk)
```

Figura 13. Definición de variables y métodos.

3.2 Fase de Pruebas

Una estrategia de prueba de software proporciona una guía que describe los pasos que deben realizarse como parte de la prueba. Debe incluir pruebas de bajo nivel, necesarias para verificar que un pequeño segmento de código se implementó correctamente, así como pruebas de alto nivel, que validan las principales funciones del sistema a partir de los requisitos del sistema. Un estrategia de prueba incluye niveles, método y técnicas de prueba (24).

3.2.1 Pruebas unitarias

Las pruebas unitarias son una de las piedras angulares de XP. Cada uno de los módulos deben pasar las pruebas unitarias antes de ser liberados o publicados (43).

3.2.2 Método de prueba de caja blanca

Las pruebas de caja blanca son las encargadas de verificar el código. Se realiza a los principales algoritmos y procedimientos. Al usar el método de caja blanca puede derivar casos de prueba que (24):

1. Garanticen que todas las rutas independientes dentro de un módulo se revisaron al menos una vez.
2. Revisen todas las decisiones lógicas en sus lados verdadero y falso.
3. Ejecuten todos los bucles en sus fronteras y dentro de sus fronteras operativas.
4. Revisen estructuras de datos internas para garantizar su validez.

3.2.2.1 Técnica de prueba de camino básico o ruta básica

La técnica del camino básico permite derivar una medida de complejidad lógica de un diseño de procedimiento y usarla como guía para definir un conjunto básico de rutas de ejecución. Los casos de prueba derivados para revisar el conjunto básico garantizan que se ejecuten al menos una vez cada sentencia durante la prueba (24).

Complejidad ciclomática

La complejidad ciclomática es una medición de software que proporciona una evaluación cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de *caminos independientes*[3] del *conjunto básico*[4] de un programa, el cual proporciona un límite superior para el número de pruebas que se deben realizarse que aseguren que se ejecuta cada sentencia al menos una vez. La complejidad ciclomática se calcula mediante las fórmulas (24):

1. $V(G) = E - N + 2$

   E: número de aristas N: número de nodos.
2. $V(G) = P + 1$

   P: número de nodos predicados (nodos de los cuales parten dos o más aristas).
3. $V(G) = R$

   R: número de regiones del grafo.

[3] **Camino independiente:** cualquier camino del programa que introduce un nuevo conjunto de sentencias de proceso o una nueva condición.

[4] **Conjunto básico:** conjunto de caminos independientes.

Para realizar la técnica de prueba del camino básico se selecciona como ejemplo el método *evaluarCriterio*, el cual se muestra en la **Figura *14*** y evalúa un criterio adicionado por el especialista en el sistema.

```
def evaluarCriterio(request, dominio="N"):
    if (request.method == 'POST' and dominio=="N"):
        form = EvaluarCriterioNumericoForm(request.POST)
        if form.is_valid():
            c = form.save(commit=False)
            c.experto = Experto.objects.get(usuario=request.user.id)
            c.save()
            form.save_m2m()
            return redirect(to='/')
    elif (request.method == 'POST' and dominio=="L"):
        form = EvaluarCriterioLinguisticoForm(request.POST)
        if form.is_valid():
            c = form.save(commit=False)
            c.experto = Experto.objects.get(usuario=request.user.id)
            c.save()
            form.save_m2m()
            return redirect(to='/')
    elif (request.method == 'POST' and dominio=="I"):
        form = EvaluarCriterioIntervalarForm(request.POST)
        if form.is_valid():
            c = form.save(commit=False)
            c.experto = Experto.objects.get(usuario=request.user.id)
            c.save()
            form.save_m2m()
            return redirect(to='/')
    elif dominio=='N':
        form = EvaluarCriterioNumericoForm()
        return render(request, 'Evaluacion/evaluarCriterioNumerico.html', {'form': form})
    elif dominio=='L':
        form = EvaluarCriterioLinguisticoForm()
        return render(request, 'Evaluacion/evaluarCriterioLinguistico.html', {'form': form})
    else:
        form = EvaluarCriterioIntervalarForm()
        return render(request, 'Evaluacion/evaluarCriterioIntervalar.html', {'form': form})
```

Figura 14.Código del método *evaluarCriterio*.

A continuación, se establecen los pasos a seguir para aplicar la técnica:

1. Representar el método en un grafo de flujo

En la **Figura *15*** se muestra el grafo de flujo resultante.

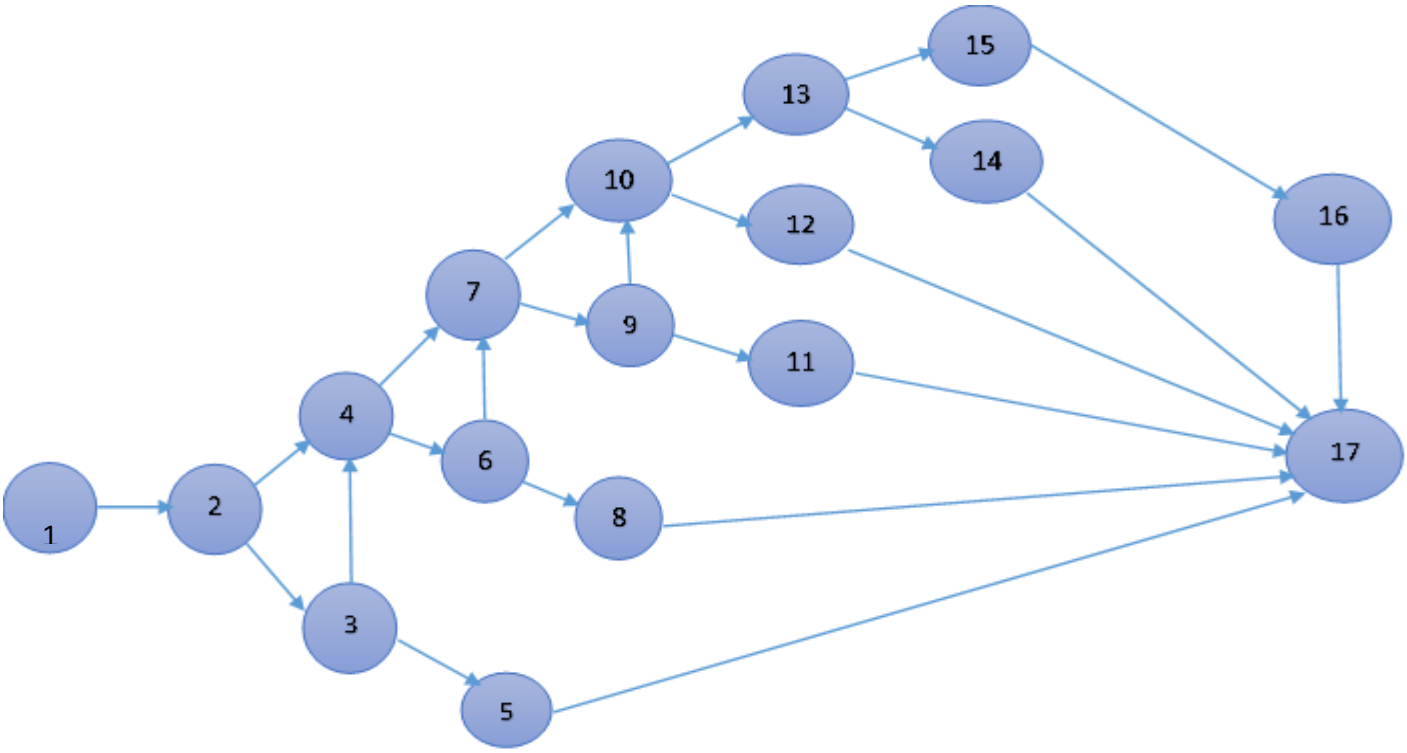


Figura 15. Grafo de flujo.

2. Determinar la complejidad ciclomática

   $V(G) = E - N + 2 = 24 - 17 + 2 = 9$

   $V(G) = P + 1 = 8 + 1 = 9$

   $V(G) = R = 9$

   Calculada la complejidad ciclomática a partir de las tres fórmulas definidas se obtiene como resultado 9, lo que determina el número de caminos independientes y el mínimo número de casos de prueba.

3. Determinar el conjunto básico de caminos independientes.

   Debido al gran número de caminos independientes resultantes solo se muestran tres de estos caminos:

   camino 1: 1-2-3-8-17

   camino 2: 1-2-3-4-6-8-17

   camino 3: 1-2-4-6-8-17

4. Preparar los casos de prueba que fuercen la ejecución de cada camino en el conjunto básico.
   Se diseñaron los siguientes casos de prueba para la ejecución de los tres caminos independientes mostrados anteriormente.

Tabla 10. Caso de prueba del camino 1

CASO DE PRUEBA CAMINO 1	
Condición de ejecución	Usuario experto autenticado en el sistema.
Entrada	request=POST dominio=N
Resultados esperados	Se muestra la interfaz para la evaluación de los criterios en el dominio numérico.

Tabla 11. Caso de prueba del camino 2

CASO DE PRUEBA CAMINO 2	
Condición de ejecución	Usuario experto autenticado en el sistema.
Entrada	request=POST dominio=L formulario no válido en el dominio numérico
Resultados esperados	Se muestra la interfaz para la evaluación de los criterios en el dominio lingüístico.

Tabla 12. Caso de prueba del camino 3

CASO DE PRUEBA CAMINO 3	
Condición de ejecución	Usuario experto autenticado en el sistema.
Entrada	request=POST dominio=I
Resultados esperados	Se muestra la interfaz para la evaluación de los criterios en el dominio Intervalar.

Análisis de los resultados

Las pruebas de caja blanca realizadas para evaluar el funcionamiento de las operaciones resultaron satisfactorias, evidenciándose así la estabilidad de la lógica aplicada al código.

3.2.3 Pruebas de aceptación

En la metodología XP, la cual rige el proceso de desarrollo de la herramienta, las pruebas de aceptación son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada. Son consideradas como *"pruebas de caja negra"*. Los clientes son responsables de verificar que los resultados de éstas pruebas sean correctos (43).

Se realizó un Caso de Prueba de Aceptación (CPA) por cada HU. Se especificaron diversos escenarios para comprobar que cada HU está implementada correctamente. La **Tabla *13*** muestra el CPA asociado a la HU Adicionar ficha técnica del proyecto.

Tabla 13. CPA de la Historia de usuario Adicionar ficha técnica del proyecto.

CASO DE PRUEBA DE ACEPTACIÓN		
Código: CPA11	**Historia de usuario:** HU11 Adicionar ficha técnica del proyecto	
Funcionalidad que se prueba: Adicionar ficha técnica del proyecto		
Condiciones de ejecución: El usuario con permisos de especialista se autentica en el sistema. En la página principal en el menú lateral izquierdo se selecciona **Proyectos**.		
Acción	**Datos de entradas**	**Resultado esperado**
El especialista selecciona la opción **Adicionar Ficha Técnica.**		El sistema muestra un formulario para introducir los datos para crear una ficha técnica de un proyecto.
EC1: El especialista introduce los datos para crear la ficha técnica del proyecto a evaluar correctamente.	Datos correctos de la ficha técnica del proyecto a evaluar.	El sistema valida que los datos sean correctos y los guarda, mostrando un mensaje indicando que los datos han sido adicionados

		satisfactoriamente. Seguidamente se muestra el listado con las fichas técnicas de los proyectos adicionadas en el sistema.
EC2: El especialista introduce los datos para crear la ficha técnica del proyecto a evaluar incorrectamente.	Datos incorrectos de la ficha técnica del proyecto.	El sistema valida los datos, al ser incorrectos muestra un mensaje indicando que los datos introducidos son incorrectos.
EC3: El especialista introduce los datos para crear una ficha técnica de un proyecto dejando campos vacíos.	Datos nulos de la ficha técnica del proyecto a evaluar.	El sistema valida los datos, al existir campos vacíos muestra un mensaje indicando que existen campos vacíos. El sistema resalta los campos vacíos.
Evaluación de la prueba: Satisfactoria		

Análisis de los resultados

La **Figura *16*** muestra los resultados obtenidos luego de aplicar las pruebas de aceptación.

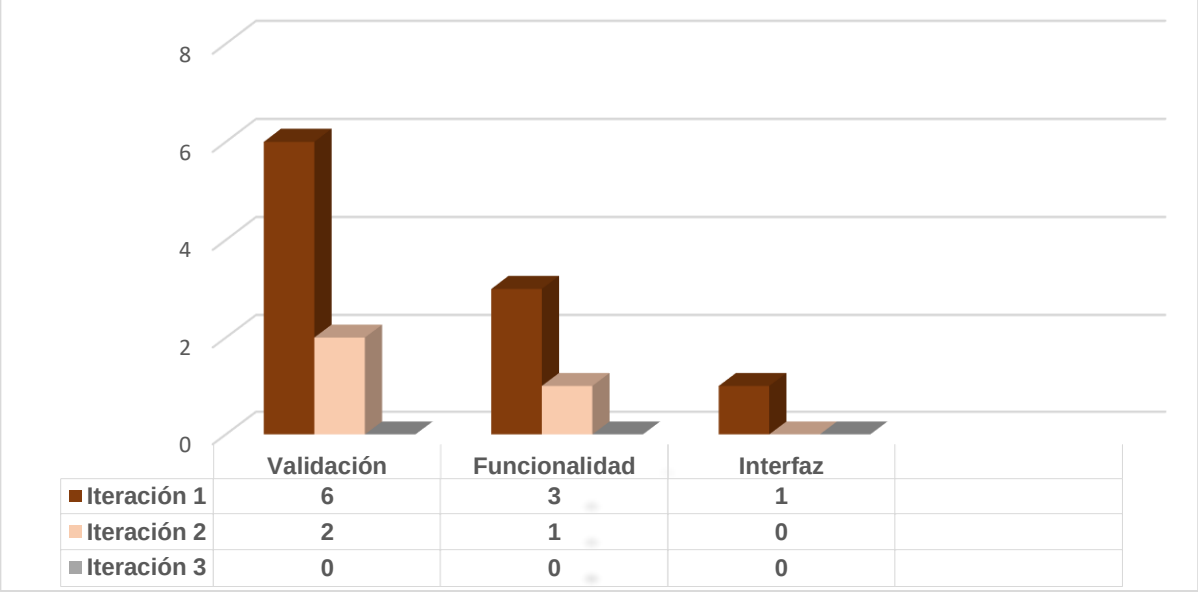


Figura 16. Resultado método de prueba caja negra. No conformidades encontradas.

Se realizaron un total de 3 iteraciones, donde se ejecutaron cada uno de los casos de prueba de aceptación diseñados para cada funcionalidad del sistema. En una primera iteración fueron encontradas 10 no conformidades separadas en 6 de validación, 3 de funcionalidad y 1 de interfaz. En la segunda iteración se resolvió la no conformidad de interfaz, se obtuvieron 2 de validación y 1 de interfaz. Finalmente fueron resueltas en una tercera iteración, obteniéndose resultados satisfactorios de la herramienta. Culminadas las pruebas y corregidas las no conformidades el cliente avaló la herramienta con la entrega del certificado de aceptación del producto.

3.3 Validación de la solución

En la validación de la solución se seleccionaron tres proyectos reales desarrollados en la UCI, de los cuales se tiene los resultados del análisis de factibilidad realizado en su concepción. A continuación, se describen cada uno de estos proyectos:

Proyecto 1: El proyecto tiene como objetivo informatizar los procesos aduanales para un mejor control y seguimiento de la organización y del país.

Proyecto 2: El proyecto tiene como objetivo informatizar el sistema penitenciario cubano, para apoyar los procesos de trabajo que se ejecutan para el control, tratamiento y atención a los internos en los establecimientos penitenciarios y los correspondientes a los niveles de mando.

Proyecto 3: El proyecto tiene como objetivo diagnosticar el estado actual de una empresa en las dimensiones de la arquitectura empresarial para conocer las principales debilidades y oportunidades existentes, determinar el estado deseado y los estados intermedios para alcanzarlo.

Para el análisis de factibilidad de los proyectos a partir del modelo lingüístico 2-tuplas se seleccionaron cinco expertos con experiencia en el desarrollo de proyectos informáticos y como líderes de equipos de proyectos. Los expertos evaluaron un total de 13 criterios. Los expertos evaluaron cada criterio eligiendo para ello uno de los tres dominios definidos. Cada criterio tiene asociado un peso según la relevancia del mismo. La **Tabla *14*** muestra los criterios utilizados y su código, donde:

C: criterio comercial T: criterio técnico S: criterio social E: criterio económico

Tabla 14. Criterios utilizados por los expertos.

Código	Criterios
C1	Grado de comercialización en el mercado
T1	Tamaño del proyecto
T2	Criticidad de la solución
T3	Conocimiento del dominio por parte del equipo de desarrollo
T4	Recursos humanos
T5	Tecnología disponible para enfrentar el proyecto
T6	Organización del modelo de producción
T7	Novedad del producto
S1	Número de personas beneficiadas
S2	Impacto en la sociedad
E1	Valor Actual Neto (VAN)
E2	Tasa Interna de Retorno (TIR)
E3	Recuperación de la inversión en el período (PRI)

Para el análisis de los resultados obtenidos por la herramienta se determina el grado de aceptación de los proyectos con las siguientes medidas propuestas por Peña (31):

- *Casos positivos*: casos donde el resultado de la herramienta y la ejecución real del proyecto coinciden.
- *Casos falsos positivos*: casos donde el resultado de la herramienta es factible y la ejecución real del proyecto fue cancelado o no exitoso.
- *Casos falsos negativos*: casos donde el resultado de la herramienta es no factible y la ejecución real del proyecto fue factible.

Las preferencias de los expertos fueron unificadas sobre el dominio lingüístico y transformadas en 2-tuplas. Se utilizó el CBTL de siete términos, representado geométricamente en la **Figura *2***.

Análisis de los resultados

Los resultados de la herramienta luego de calculada la factibilidad global de cada proyecto se ofrecen en la última columna de la **Tabla *15***, donde se evidencia la factibilidad arrojada por la herramienta y la precisión del resultado.

Tabla 15. Evaluación global de los proyectos.

Proyectos	Criterios	Evaluación de los proyectos (factibilidad, precisión)
p_1	T1	(FA; 0.31)
	T2	
	T3	
	T4	
	T5	
	C1	
	T6	
	T7	
	S1	

	S2	
	E1	
	E2	
	E3	
p_2	T1	(FA; -0.14)
	T2	
	T3	
	T4	
	T5	
	C1	
	T6	
	T7	
	S1	
	S2	
	E1	
	E2	
	E3	
p_3	T1	(FM; -0.15)
	T2	
	T3	
	T4	
	T5	
	C1	
	T6	
	T7	

	S1	
	S2	
	E1	
	E2	
	E3	

La información arrojada por la herramienta puede ser analizada de diferentes formas dada la naturaleza de los proyectos y las condiciones de la entidad desarrolladora, y así tomar decisiones en correspondencia con las características propias de cada proyecto y los resultados obtenidos. En un inicio pueden ser interpretados los resultados de la evaluación global de los proyectos y en caso de ser necesario refinar el análisis haciendo énfasis en algunos criterios específicos. El orden según la factibilidad global se obtuvo luego de aplicado los operadores de comparación para 2-tuplas definidos en el *epígrafe 1.3.2*, donde $\{p_1 > p_2 > p_3\}$ ya que $\{(\mathrm{FA};\ 0.31) > (\mathrm{FA};\ -0.14) > (\mathrm{FM};\ -0.15)\}$. Como resultado final del análisis de factibilidad realizado mediante la herramienta se obtuvo como grado de aceptación un caso positivo, ya que los tres proyectos resultaron factibles, lo que coincide con los resultados reales de los análisis realizados antes de su inicio. La validación de la solución evidenció que la herramienta desarrollada contribuye a la toma de decisiones en los análisis de factibilidad de proyectos de software, con tratamiento en la incertidumbre de la información.

Conclusiones parciales

Al concluir el capítulo se llegó a las siguientes conclusiones:

- Los casos de prueba diseñados para las pruebas unitarias y de aceptación garantizan la verificación y validación de la herramienta desarrollada para análisis de factibilidad proyectos de software.
- La herramienta informática brinda facilidades a los especialistas principales de proyectos para decidir la ejecución o no de un proyecto.
- La validación de la solución demuestra que la herramienta desarrollada contribuye al tratamiento de la incertidumbre en los análisis de factibilidad de proyectos de software para la toma de decisiones.

CONCLUSIONES

Finalizada la investigación se puede concluir que se cumplieron todos los objetivos propuesto, resaltando que:

- A partir de la revisión de la literatura asociada con el objeto de estudio, se concluye que los análisis de factibilidad de proyectos de software pueden tratarse como un problema de toma de decisiones, con criterios de evaluación que manejan información heterogénea, evaluados por expertos con diferentes grados de experticia.
- Las herramientas existentes para los análisis de factibilidad de proyectos de software poseen limitantes, ya que algunas no consideran la incertidumbre presente en la información y otras solo tienen en cuenta criterios económicos.
- El uso de técnicas de softcomputing permitió obtener mejores resultados en la toma de decisiones en los análisis de factibilidad.
- Las fases de la metodología de software XP permitieron la obtención de los artefactos que guiaron el proceso de desarrollo de la herramienta.
- La herramienta informática obtenida contribuye al tratamiento de la incertidumbre en los análisis de factibilidad de proyectos de software para la toma de decisiones.

RECOMENDACIONES

Se propone el uso de sistemas inteligentes para estimar los riesgos que se pueden desprenden del análisis de factibilidad arrojado por la herramienta.

REFERENCIAS BIBLIGRÁFICAS

1. **Serra, C.E.M. and Kunc, M.** *Benefits realisation management and its influence on project success and on the execution of business strategies.* s.l. : International Journal of Project Management, 2015. pp. 53-66. Vol. 33.

2. **Andrade, M.J. Villagrán.** *Estudio de factibilidad para la implementación de un centro de interpretación del patrimonio cultural y natural de Yaharcocha, cantón Ibarra, provincia de Imbabura.* 2015.

3. **Echeverría, L.** *Contribución de la tecnología en la gestión del conocimiento entre los grupos de investigación del área de informática.* 2017. pp. 21-28. Vol. 6.

4. **STANDISH-GROUP.** *Chaos Manifiesto 2015.* [ed.] The Standish Group. 2015.

5. **Burstein, F. and Holsapple, C.W.** *Handbook on Decision Support Systems 1.* Springer-Verlag Berlin Heidelberg : s.n., 2008.

6. **Tanaka, Á.T. and Montero, C.M.C.** *Valorización de opciones reales: modelo Ornstein-Uhlenbeck.* 52-62. s.l. : Journal of Economics, Finance and Administrative Science, 2016. Vol. 21.

7. **Arza, L.** *Modelo computacional para la recomendación de roles en el proceso de ubicación de estudiantes en la industria de software.* La Habana : s.n., 2013.

8. **Torres, S.** *Modelo de evaluación de competencias a partir de evidencias durante la gestión de proyectos.* Universidad de las Ciencias Informáticas. : s.n., 2015.

9. **Zhang, X.** *A prototype system dynamic model for assessing the sustainability of construction projects.* International Journal of Project Management : s.n., 2014. pp. 66-76.

10. **Silvius, A.G. and Schipper, R.** *Exploring the relationship between sustainability and project success-conceptual model and expected relationships.* IJISPM-INTERNATIONAL JOURNAL OF INFORMATION SYSTEMS AND PROJECT MANAGEMEN : s.n., 2016. pp. 5-22. Vol. 4.

11. **Aarseth, W.** *Project sustainability strategies: a systematic literature review.* International Journal of Project Management : s.n., 2017. pp. 1071-1083. Vol. 35.

12. **Elio Cables, M.S.G.-C. and Teresa Lamata, M.** *he LTOPSIS: An alternative to TOPSIS decision-making approach for linguistic variables.* Expert Systems with Applications : s.n., 2012. pp. 2119-2126. Vol. 39.

13. **Chen, C.T., Hung, W.Z. and Cheng, H.L.** *Cheng, Applying linguistic PROMETHEE method in investment portfolio decision-making.* International Journal of Electronic Business Management : s.n., 2011. p. 139. Vol. 9.

14. **BoraSystems.** EasyPlanEx. [Online] 2017. https://www.borasystems.com/es/productos-software/easyplanex/.

15. **Choice, Expert.** ExpertChoice. [Online] 2017. https://www.expert-choice.com/.

16. **Decide.** ITESO. [Online] 2013. http://www.iteso.mx/web/general/detalle?group_id=57482.

17. **EvalAs.** El sitio agrícola. [Online] 02 03, 2015. http://www.elsitioagricola.com/articulo.aspx?id=40&nombre=EvalAs%3A+Software+de+Evaluaci%C3%B3n+de+Proyectos.

18. **Intecplan.** intecplan. [Online] 2018. https://www.intecplan.com.mx/.

19. **Bolaños, R and Correa, C.** *Planeamiento de la transmisión considerando seguridad e incertidumbre en la demanda empleando programación no lineal y técnicas evolutivas.* 2014. pp. 62-76.

20. **Wang, W.P. and Tang, M.C .** *A Multi-criteria Assessment for R&D Innovation with Fuzzy Computing with Words. in Modelling, Computation and Optimization in Information Systems and Management Sciences.* 2015.

21. **Bemúdez, A., Lugo, J.A. and Piñero, P.Y.** *An Adaptive-Network-Based Fuzzy Inference System for Project Evaluation.* 2015. pp. 299-313.

22. **Melo Rodríguez, J.A. and Cortés, C.A.** *Análisis de vulnerabilidad de sistemas de potencia incluyendo incertidumbre en las variables con lógica difusa tipo 2.* 2016. pp. 100-119.

23. **Salazar, Ivarth Palacio.** *Guía práctica para la identificación, formulación y evaluación de proyectos.* Primera. Bogotá : Universidad del Rosario, 2010. 978-958-738-067-5.

24. **Pressman, Roger S.** *Ingeniería del software. Un enfoque práctico.* Séptima edición. México : s.n., 2010. 978-607-15-0314-5.

25. **Sommerville, Ian.** *Ingeniería de Software.* Novena edición. México : s.n., 2011. p. 37. 978-607-32-0603-7.

26. **LACRUZ, BLANCA MARQUINA.** *GUÍA PRÁCTICA PARA LA PRESENTACIÓN Y ELABORACIÓN DE ESTUDIOS DE FACTIBILIDAD.* 2010.

27. **Miranda, Juan José Mirando.** *GESTION DE PROYECTOS: IDENTIFICACION - FORMULACIONEVALUACIÓN-FINANCIERA–ECONÓMICA–SOCIAL–AMBIENTAL.* Cuarta edición. 2005.

28. **Almaguer, Daniarys Ramírez.** *Etapas del análisis de factibilidad. Compendio bibliográfico. .* 2009.

29. **Zadeh, L.** *Fuzzy logic = computing with words.* s.l. : IEEE Transactions on Fuzzy Systems, 1996. pp. 103-111.

30. **Herrera, F., Martínez, L. and Sánchez, P.J.** *Managing non-homogeneous information in group decision making.* España : European Journal of Operational Research, 2005.

31. **Abreu, Marieta Peña.** *Modelo para análisis de factibilidad de proyectos de software en entornos de incertidumbre.* 2017.

32. **Herrera, Francisco and Martinez, Luis.** *A 2-tuple fuzzy linguistic representation model for computing with words.* s.l. : IEEE Transactions of Fuzzy Systems, 2000. pp. 746-752. Vol. 8.

33. **Figueroa, Roberth G., Solís, Camilo J. and Cabrera, Armando A.** *Metodologías tradicionales vs. metodologías ágiles.* 2010.

34. ***Molina Ríos, Jimmy Rolando, et al.*** *4, 2016, Revista Latinoamericana de Ingeniería de Software, Vol. 4.*

35. ***OMG.*** *OCUP 2 - OMG Certified UML Professional. [Online] Object Management Group. [Cited: 03 27, 2018.] http://www.omg.org/ocup-2/index.htm/.*

36. ***Jacobson, I., Booch, G. and Rumbaugh, J.*** *El Proceso Unificado de Desarrollo. s.l. : Addision Wesley, 2000.*

37. ***Paradigm, Visual.*** *Knowledge Reuse Group. [Online] 2015. http://www.ie.inf.uc3m.es/grupo/docencia/reglada/Is1y2/PracticaVP.pdf.*

38. ***SGBD.*** *Desarrollo Web. [Online] julio 31, 2007. http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.*

39. ***Almeira, Adriana Sandra and Pérez, V.*** *Arquitectura de Software: Estilos y Patrones. 2007.*

40. ***MTV.*** *LibrosWeb. [Online] 2018. http://librosweb.es/libro/django_1_0/capitulo_5/el_patron_de_diseno_mtv.html.*

41. ***Larman, Craig.*** *UML y Patrones. Introducción al análisis y diseño orientado a objetos. [ed.] Pablo Eduardo Roig Vázquez. [trans.] Luz María Hernández Rodríguez. Primera. México : PRENTICE HALL, 1999. pp. 189-209. ISBN: 970-17-0261-1.*

42. ***Christiansson, Benneth.*** *GoF Design Patterns-with examples using Java and UML2. 2008.*

43. ***Joskowicz, José.*** *Reglas y prácticas en eXtreme Programing. 2008.*

44. ***Escalona, María José and Koch, Nora.*** *Ingeniería de Requisitos en Aplicaciones para la Web. Un estudio comparativo. España : s.n., 2002.*

45. ***Sommerville, Ian.*** *Ingeniería del Software. Novena Edición. México : s.n., 2011. 978-607-32-0603-7.*

46. ***Jeffries, R., Anderson, A. and Hendrickson, C.*** *Extreme Programming Installed. 2001.*

47. ***Silberschatz, Abraham , Korth, Henry F. and Sudarshan, S.*** *FUNDAMENTOS DE BASES DE DATOS. Cuarta. 2002. pp. 19-48. 0-07-228363-7.*

48. ***Rumbaugh, James, Jacobson, Ivar and Booch, Grady.*** *El Lenguaje Unificado de Modelado. Manual de referencia. [trans.] Addison Wesley Longman. Madrid. España : Pearson Educación, 2000. ISBN: 84-7829-037-0.*