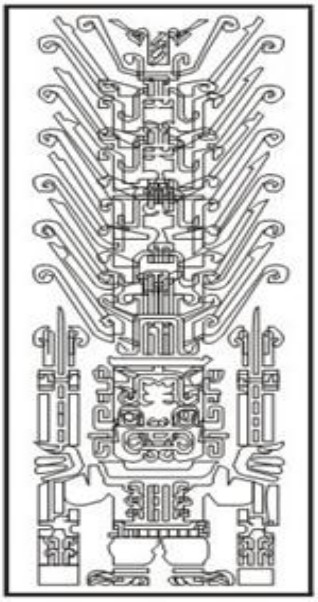
UNIVERSIDAD NACIONAL FEDERICO VILLARREAL

ESCUELA UNIVERSITARIA DE POST GRADO



PROYECTO DE PLAN DE TESIS PARA OBTENER EL GRADO ACADÉMICO DE DOCTOR

TÍTULO:

"*METODOLOGÍA INTEGRAL BASADA EN EL DESCUBRIMIENTO DE CONOCIMIENTO DE DATOS QUE PERMITA LA ESTIMACIÓN FUNCIONAL PARA MEJORAR LA GESTIÓN DE PROYECTOS DE SOFTWARE*"

Graduando:

PEDRO MARTIN LEZAMA GONZALES.

Lima - Perú

INDICE

TÍTULO:

Metodología integral basada en el descubrimiento de conocimiento de datos que permita la estimación funcional para mejorar la gestión de proyectos de *software*.

AUTOR

Pedro Martin Lezama Gonzales

LUGAR DONDE SE VA A REALIZAR LA INVESTIGACIÓN:

1. Edelnor: Jr. César López Rojas # 201 Urb. Maranga San Miguel.
2. Stefanini: Av. Larco 880 Miraflores, 4to Piso

I. DESCRIPCIÓN DEL PROYECTO

I.1. ANTECEDENTES

La medición cuenta con una larga tradición y constituye una disciplina fundamental en cualquier ingeniería, y la ingeniería del *software* no debe ser una excepción.

La medición *software* es una disciplina joven, y ello ha influido notablemente en que la ingeniería del *software* no haya alcanzado aún el grado de madurez que tienen otras ingenierías.

Sin embargo, en la actualidad pocos dudan de la importancia de la medición para incrementar la calidad, la productividad en el desarrollo y mantenimiento del *software*. La necesidad y motivación por medir se ha incrementado notablemente con la preocupación de las organizaciones por alcanzar mayores niveles de madurez y las consiguientes certificaciones basadas en modelos y normas como:

- *ISO* 9000 Normas sobre calidad y gestión de calidad.

- *ISO* 15504 Determinación de la capacidad de mejora del proceso de *software.*

- *CMMI* integración de modelos de madurez de capacidades.

- La organización *ISO/IEC* ha definido un estándar de medida del tamaño funcional, titulado *'ISO/IEC* 14143-1:1998', revisado en '*ISO/IEC* 14143-1:2007'. Con base en este estándar se han declarado, como métodos estándares de recuento, los siguientes:

    - *ISO/IEC* 20968:2002 *Mk II Function Point Analysis - Counting Practices Manual.*

    - *ISO/IEC* 24570:2005 *NESMA Guide to Using Function Point Analysis.*

    - *ISO/IEC* 20926:2009 *IFPUG 4.3.1 Unadjusted functional size measurement method - Counting practices manual*. (A partir de este momento denominado *UFP*).

    - *ISO/IEC* 19761:2011 *COSMIC-FFP - A Functional Size Measurement Method.*

Numerosos estudios con respecto al tema de la estimación de proyectos se han publicado junto con diferentes herramientas y análisis de diversas metodologías que intentan establecer algún tipo de control, de acuerdo con la técnica utilizada, sobre las actividades

definidas por un previo proceso de recolección y análisis de requisitos y cuya realización implica considerar costos, calendario y esfuerzo humano. Algunos de estos estudios se describen brevemente a continuación:

I.1.1. TRATADOS Y TÉCNICAS DE ESTIMACIÓN

1. (Peters, 1999, pág. 1). Considera cuatro pasos básicos en la estimación de proyectos de *software*:

   I. Líneas de código (*LOC*), puntos de función y otras posibles unidades de medida.

   II. Estimación del esfuerzo ya sea en: personas/meses o personas/horas.

   III. Estimación del calendario.

   IV. Estimación del costo del proyecto.

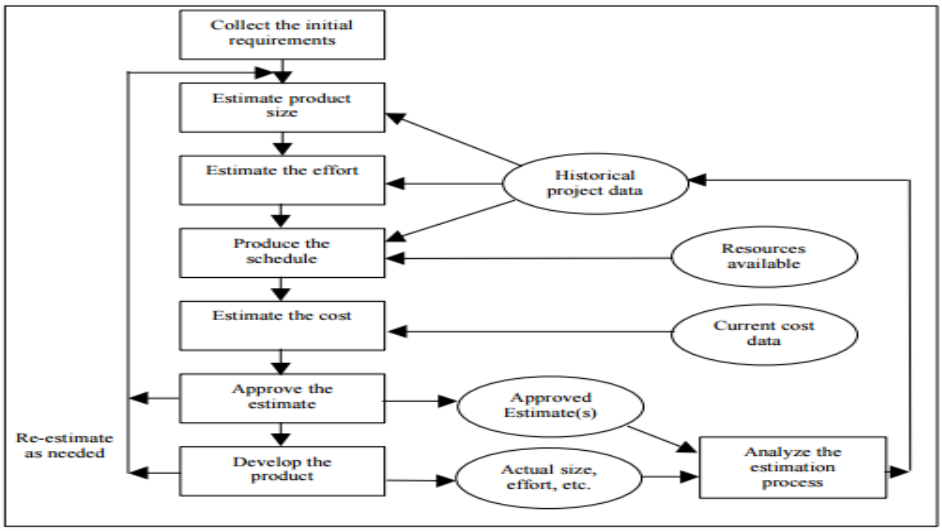


Imagen 1: Proceso básico de estimación de proyectos de Software. Fuente: (Peters, 1999, pág. 3).

2. (Labdelaoui, 2001) Desarrolló un análisis exhaustivo acerca de los métodos de estimación de esfuerzo más importantes que existían en ese momento, en su libro "*Desarrollo de Métodos de Estimación de Tamaño Funcional del Software Aplicados a Enfoques*". Este trabajo describió los pasos que los diferentes métodos de medición de tamaño funcional adoptan, con el fin de calcular el tamaño del *software* y el esfuerzo en proyectos de este mismo tipo. Los métodos descritos son:

   - *IFPUG FPA.*
   - *Feature Points (FP).*
   - *Mark II FPA.*
   - *Full Function Points v1.0 (FFP).*
   - *COSMIC FFP v2.0.*
   - Puntos de casos de uso (*UCP*).
   - *Predictive Object Points (POP).*

3. (Gómez, Guía Práctica de Estimación y Medición de Software, 2014) en el libro "*Guía Práctica de Estimación y Medición de Proyectos Software*" realiza un estudio detallado de las técnicas de medición, entre ellas se pueden mencionar:

   - Método de estimación de 3 puntos.

- Método de técnicas *Delphi*.
- Método de *COCOMO* 81.
- Método de *COCOMO* 2000.
- Método de *Putnam o SLIM*.
- Método de *Planning Poker*.
- Método de *IFPUG FPA*.
- Método de *NESMA*.
- Método de *MKII-FPA*.
- Método de *COSMIC FFP*.
- Método de *FiSMA*.
- Método de estimación temprana de *NESMA*.
- Método de *SiFP*.
- Método de puntos función 3D.
- Método de puntos característica (*Feature Points*).
- Método de puntos objeto.
- Método de puntos casos de uso.
- Método de lógica difusa (*Fuzzy Logic*).

- Método de catálogo de componentes (*Standard Components*).
- Método de puntos de historia.
- Método de *T-SHIRT Sizing.*
- Método de *SNAP*.

I.1.2. TESIS DOCTORALES

1. (Aroba Páez, 2003) en la tesis doctoral "*Avances en la Toma de Decisiones en Proyectos de Desarrollo de Software*", propuso nuevas contribuciones en el campo de la toma de decisiones en los proyectos de desarrollo de *software* (a partir de este momento denominado *PDS*), que permitan obtener información de tipo cualitativo sobre el funcionamiento de los *PDS*, además realiza un análisis comparativo e implementación de técnicas de predicción que pueden ser aplicadas en el ámbito de los *PDS*. La información analizada son las lecciones aprendidas de un conjunto de proyectos, empleando la minería de datos, de tal forma que el director del proyecto pueda tomar decisiones lo más asertivas posible.

2. (Aranda, 2008) en la tesis doctoral "*Marco para la Elicitación de Requisitos Software en Procesos de Desarrollo Global",* propuso una serie de etapas que mejoren la comunicación durante la elicitación en entornos distribuidos y como consecuencia, obtener requisitos más precisos. Empleando conceptos innovadores en esta etapa de la ingeniería del *software* que pertenecen a un área de investigación llamada

informática cognitiva, disciplina que se centra en el estudio del procesamiento de la información en el cerebro, especialmente respecto a actividades como adquisición, representación, y comunicación de la información. Además analiza una amplia cantidad de métodos y técnicas de elicitación existentes, y en buscar literatura relacionada al proceso de selección de las técnicas a utilizar.

El modelo propuesto es:

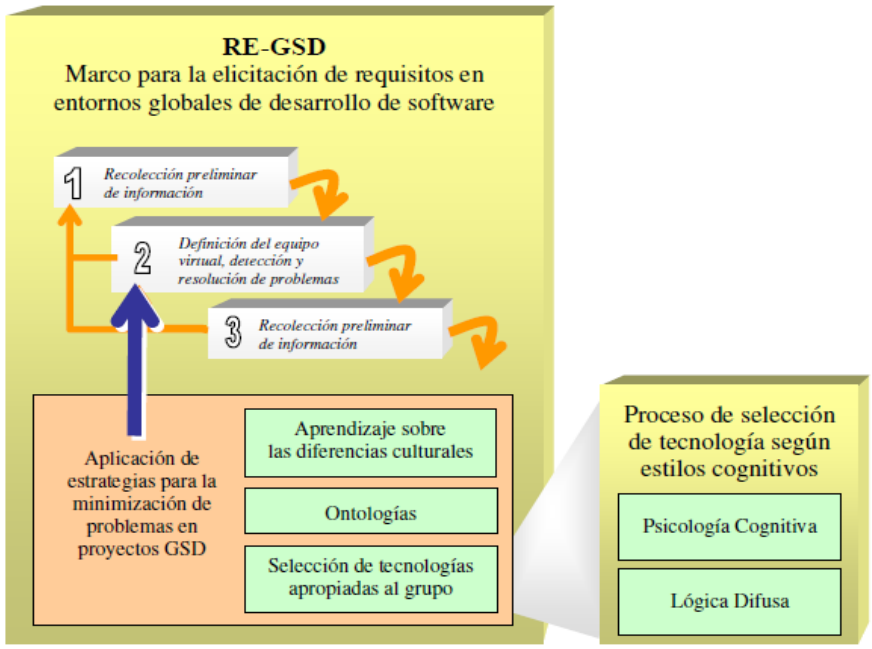


Imagen 2: Componentes del modelo para la elicitación de requisitos en entornos globales. Fuente: (Aranda, 2008).

3. (Matturro Mazoni, 2010) en la tesis doctoral "*Modelo Para la Gestión del Conocimiento y la Experiencia Integrada a las Prácticas y Procesos de Desarrollo Software*", propuso un modelo para la gestión del conocimiento y la experiencia cuyas fases y tareas, se integran a las actividades de los proyectos de *software*, así como a las de mejora de las

prácticas y procesos en uso en una organización. Además propone también una herramienta para capturar los conocimientos y experiencias en forma simultánea a la realización de las actividades de proyecto, que se diferencia de los modos tradicionales basados en el análisis post mortem de proyectos y técnicas similares.

El modelo propuesto es denominado "***ele***" que consta de 8 fases:

Imagen 3: Componentes del modelo "***ele".*** Fuente: (Matturro Mazoni, 2010).

El esquema funcional propuesto es:

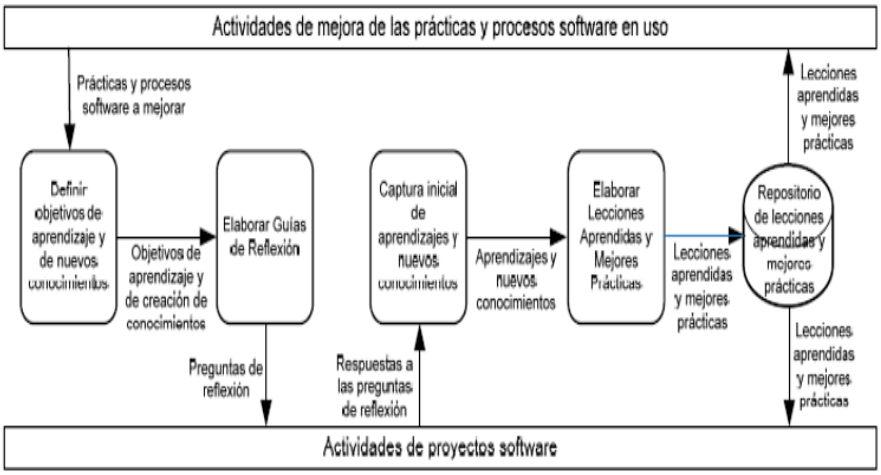


Imagen 4: Esquema funcional de la tesis. Fuente: (Matturro Mazoni, 2010)

4. (Bertolami, 2010) en la tesis doctoral "*Estimación del Tamaño Funcional del Software en la Elicitación de Requerimientos*", planteó un procedimiento *scenario function points* (a partir de este momento denominado *SFP*) que permite estimar el tamaño funcional de un sistema de *software* a partir de un producto de la elicitación de requerimientos, específicamente, los escenarios generados de acuerdo al enfoque de *Leite*. *SFP* está basado en el método estándar *IFPUG-FPA* y su definición está soportada por un conjunto de reglas que establecen la asociación entre los conceptos del modelo de escenarios y los propios de dicho método. Para guiar la aplicación de *SFP* esta tesis diseñó un proceso y elaboró una especificación formal. El objetivo de esta última es facilitar la capacitación y comunicación de los usuarios, favorecer la efectiva ejecución del proceso y propiciar la repetibilidad de los resultados.

5. (Pow Sang Portillo, 2012) en la tesis doctoral "*Técnicas para la estimación y planificación de proyectos de software con ciclos de vida*

incremental y paradigma orientado a objetos", Propone una técnica denominada "*tupuy*" y está conformada por tres técnicas: *UML$_2$FP*, Use *Case Precedence Diagram* (*UCPD*) e *Incremental-FP*.

- La técnica *UML$_2$FP* permite realizar el cálculo de *PFSA*. Para ello, emplea el diagrama de clases de análisis, el cual también es conocido en otras metodologías como modelo de dominio. También, utiliza las especificaciones de caso de uso.

- Para la técnica *UCPD* se emplea la especificación de los casos de uso para definir la secuencia de construcción de casos de uso.

- Finalmente, la técnica *Incremental-FP* utiliza los resultados obtenidos de emplear las técnicas *UCPD* y *UML$_2$FP*, la cual comprende la ejecución de cuatro actividades. Con estas actividades se definen los incrementos a construir, se determinan qué casos de uso se van a desarrollar en cada incremento y se estima el esfuerzo que se requiere para concluir cada incremento.

Actividades y técnicas de *Tupuy*

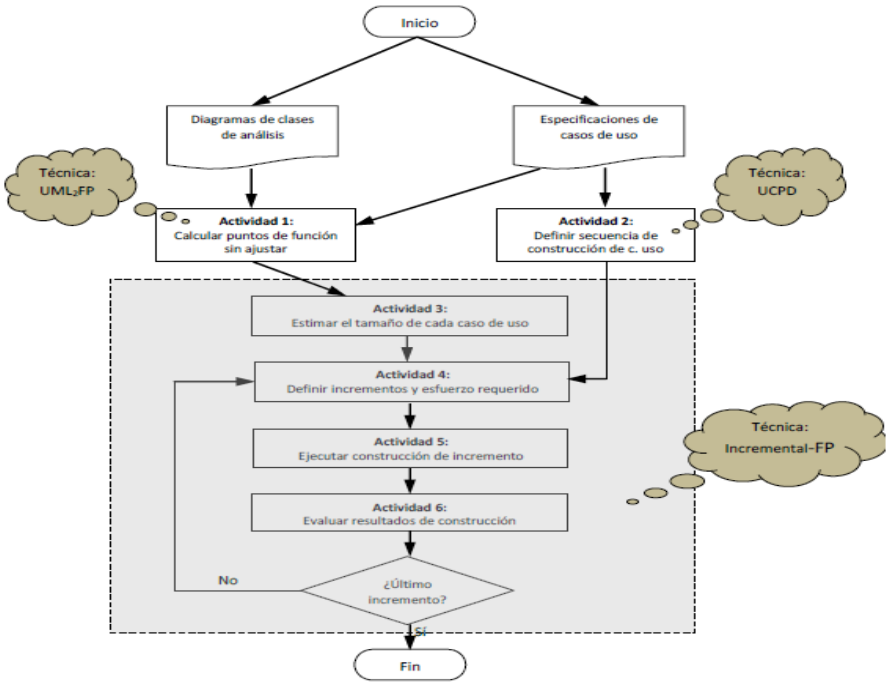


Imagen 5: Técnicas que propone *Tupuy* en nubes y las actividades que comprende cada técnica. (Pow Sang Portillo, 2012).

I.1.3. CUERPOS DE CONOCIMIENTO

1. La guía del *PMBOK*® (PMI_PMBOKv5, 2013) estableció 2 áreas de conocimiento de tiempo y costos, el primero incluye los procesos requeridos para gestionar la terminación en plazo del proyecto, el segundo incluye los procesos relacionados con planificar, estimar, presupuestar, financiar, obtener financiamiento, gestionar y controlar los costos de modo que se complete el proyecto dentro del presupuesto aprobado. Además considera un componente organizacional muy importante en los activos de los procesos de la organización (*organizational process assets*) empleado como entrada en forma específica de 38 procesos de los 47 procesos establecidos en la quinta versión (a partir de este momento denominado *APO*). Los *APO* son los

planes, los procesos, las políticas, los procedimientos y las bases de conocimiento de las lecciones aprendidas de todas las áreas de conocimiento. Recomienda además, que un proyecto no se puede establecer como cerrado, mientras no se han actualizado las lecciones aprendidas en la base de datos (a partir de este momento denominado *BD*) de conocimiento de la organización, a pesar de que el cliente pudiese haber aceptado el producto, entregable o resultado del proyecto.

I.1.4. PROCESOS DE ESTIMACIÓN

A continuación se describen los principales procesos de estimación que han venido siendo utilizados hasta el momento:

- Proceso de *Boehm.*
- Proceso de *Bailey-Basili.*
- Proceso de *DeMarco.*
- Proceso de *Heemstra.*
- Proceso de *Arifoglu.*
- Proceso de *Humphrey* (PROBE).
- *Wide band Delphi.*
- *Estimeeting.*

1. Proceso de *Boehm.*

(Boehm., 1981). Propuso un proceso de siete pasos, para estimar el coste del *software*:

- Establecer objetivos: El primer paso es determinar nuestros objetivos en la estimación. Para ello hemos de evitar gastar tiempo en recopilar información y realizar estimaciones que no sean relevantes para las decisiones que tenemos que tomar.

- Planificar recursos y datos requeridos: En este paso, debemos planificar nuestra actividad de estimación, a fin de evitar la tentación de preparar una estimación con demasiada antelación.

- Fijar los requerimientos *software*: En este tercer paso determinaremos si las especificaciones en las que están basadas nuestra estimación son económicamente viables. Necesitamos saber qué estamos intentando construir a fin de estimarlo todo con precisión.

- Detallar tanto como sea posible: El cuarto paso nos indica que hacer una estimación con mucho detalle, mejora su precisión.

- Utilizar varias técnicas y fuentes independientes: *Boehm* clasifica las técnicas de estimación como modelos algorítmicos, opiniones de los expertos, analogía, *Parkinson, price-to-win, top-down y botton-up.* Este paso nos indica que hemos de usar más de una técnica para realizar una estimación del coste. El uso de una combinación de técnicas nos permite evitar los puntos débiles de cada técnica, así como tomar lo mejor de cada una.

- Comparar e iterar estimaciones: Una vez realizada nuestra estimación utilizando dos o más técnicas, debemos comparar los resultados. Si son inconsistentes, nuestro siguiente paso es investigar las diferencias. El objetivo es converger en una estimación lo más realista posible, más que establecer un compromiso arbitrario entre las estimaciones iniciales.

- Continuación: El paso final en el proceso de estimación del coste continúa la estimación realizada en pasos anteriores. Debemos usar los resultados de la comparación entre la estimación y el valor real para mejorar nuestra técnica de estimación y nuestros modelos. Deberíamos re-estimar el coste cuando el alcance del proyecto cambia.

2. Proceso de *Bailey-Basili.*

(Bayley & Basili, 1981). Propusieron el "*Meta Modelo de Bailey-Basili*" como un proceso para estimar el esfuerzo. El proceso tiene tres pasos principales:

- Calibrar el modelo de esfuerzo: El modelo de esfuerzo se calibra usando una regresión por mínimos cuadrados, sobre un conjunto de datos locales para calcular los coeficientes del modelo.

- Estimar los límites superior e inferior del modelo de predicción: Se trata de calcular el error estándar de estimación (*Standard Error of the Estimate, SEE*). *SEE* se puede utilizar para calcular los límites superior e inferior del intervalo de confianza construido para el modelo, asumiendo una distribución normal.

- Ajustar las predicciones del modelo para tener en cuenta la significancia de los atributos conductores del coste: El efecto de los conductores del coste sobre el esfuerzo se estima calculando un factor de ajuste del esfuerzo y aplicándolo a la estimación de esfuerzo inicial. El factor de ajuste para un punto de la regresión original es el número mediante el que el valor efectivo ha de ser multiplicado para obtener el valor predicho. Un modelo de ajuste de esfuerzo se calibra utilizando regresión lineal múltiple para

determinar los coeficientes conductores del coste para cada uno de los atributos conductores.

La estimación final se realiza utilizando ambos modelos: el primero para realizar una estimación inicial del esfuerzo, y el segundo para ajustar la estimación, teniendo en cuenta los conductores del coste.

3. Proceso de *DeMarco.*

(DeMarco, 1982). Propone un proceso para desarrollar y aplicar modelos de coste basados en datos adquiridos localmente. Él defendía el uso de modelos de coste de factores simples derivados de la regresión lineal. Un modelo de coste de factores simples predice el esfuerzo para todo o parte del proyecto, basándose en una variable independiente. El esfuerzo estimado obtenido desde un modelo de coste de factores simples, se ha de ajustar aplicando un factor de corrección. Los factores de corrección, se derivan del mismo modo al propuesto por *Bailey y Basili.*

El modelo de *DeMarco* soporta estimación *botton-up* y *top-down*. Sugiere dividir el proyecto en componentes de coste, tales como: esfuerzo de diseño, esfuerzo de implementación y esfuerzo de integración. Cada componente de coste es estimado mediante uno o más modelos basados en la medida, los cuales están disponibles en el momento de realizar la estimación.

DeMarco también defendía el uso de modelos sensibles al tiempo, los cuales relacionan el esfuerzo total del proyecto con la duración. Sugiere

que se use el modelo *COCOMO* para este propósito. A continuación se indican las actividades llevadas a cabo para utilizar este modelo:

- Seleccionar el coste del componente a modelar: El primer paso del proceso implica la selección de los componentes sobre los que se va a realizar una estimación de coste. El coste de los componentes está basado en una descomposición del sistema de desarrollo en actividades, tales como especificación, diseño, implementación y test.

- Seleccionar las medidas desde las cuales predecir los componentes y el coste total: Una vez elegidos los componentes, el siguiente paso es determinar las medidas desde las cuales, el esfuerzo para completar esos componentes pueda ser predicho.

- Desarrollar los modelos de coste basados en un simple valor, basándose en los datos existentes: Una vez que se han determinado las métricas, deben desarrollarse los modelos de regresión mediante los que se predice el esfuerzo.

- Estimar el coste y el error estándar de la estimación para cada componente del modelo: Cuando el valor de las métricas utilizadas en el modelo de coste basado en un factor simple puede ser estimado o medido, el modelo se utiliza para predecir el esfuerzo de los componentes. Si se utiliza una estimación del valor de las métricas de entrada, el esfuerzo debería ser reestimado cuando haya sido calculado el valor real de tales métricas.

- Ajustar las estimaciones basadas en diferencias entre el proyecto actual y proyectos pasados: *DeMarco* indica que la estimación del esfuerzo debe ser ajustada mediante una evaluación subjetiva de las diferencias entre el proyecto en curso y los proyectos pasados.

- Calcular el coste total a partir de los costes de los componentes: Una vez estimado el coste de todos los componentes, se calcula el coste total como la suma de éstos.

- Calcular el error de estimación global: El error en la estimación total se calcula sumando el error estándar de la estimación de cada componente aislado.

- Utilizar un modelo de coste sensible al tiempo para restringir la estimación total del esfuerzo del proyecto: El paso final en el proceso de *DeMarco* utiliza la estimación del esfuerzo como entrada a un modelo sensible al tiempo, tal como *COCOMO,* para estimar la duración del proyecto.

4. Proceso de *Heemstra.*

   (Heemstra, 1992) Describe un proceso general de estimación de costes. Su proceso asume el uso de modelos de esfuerzo dependientes del tamaño, y el uso de atributos conductores del coste, unidos a estos modelos para realizar ajustes de productividad. Este proceso se divide en siete pasos:

   - Crear una base de datos de proyectos finalizados: El paso inicial en el proceso de estimación es la recolección de datos locales para validar y calibrar un modelo.

   - Validar el modelo de estimación: El entorno en el que ha sido desarrollado el modelo de coste y los proyectos finalizados en los que está basado, pueden diferir del entorno en el que el modelo es usado. *Heemstra* advierte, por consiguiente, que antes de utilizar un modelo precalibrado por primera vez en una organización, éste

debería ser validado, evaluando su precisión sobre datos de proyectos ya finalizados en la organización.

- Calibrar el modelo de estimación: Si en el paso previo se indica que el modelo no es válido para el entorno en el cual va a ser utilizado, el modelo necesita ser recalibrado usando datos de proyectos finalizados en el entorno actual.

- Estimar el tamaño: En este paso, se calcula el tamaño del *software* partiendo de las características propias del *software* a desarrollar.

- Estimar el esfuerzo y la productividad: Una vez estimado el tamaño, el siguiente paso convierte el resultado obtenido en estimación de esfuerzo. Los atributos conductores del coste con influencia en el esfuerzo del desarrollo del *software* son evaluados y utilizados para ajustar la estimación de dicho esfuerzo.

- Distribuir el esfuerzo a lo largo de las fases de realización del proyecto: En este paso, el esfuerzo total y la duración del proyecto son distribuidas a lo largo de las fases de desarrollo del *software*.

- Analizar la sensibilidad de la estimación y los riesgos asociados: En este paso se evalúa la sensibilidad del coste estimado para valores de los atributos conductores del *software*, y se determinan los riesgos asociados con la estimación de costes del proyecto.

5. Proceso de *Arifoglu.*

(Arifoglu, 1993) Propone un proceso de estimación que consta de cuatro pasos:

- Estimación del tamaño: Este paso es equivalente a la estimación de tamaño del proceso de *Heemstra*.

- Estimación del esfuerzo y duración: Este paso convierte la estimación del tamaño en estimación del esfuerzo y estimación de la duración. *Arifoglu* sugiere el uso del modelo *COCOMO* en este paso.

- Distribución del esfuerzo y la duración durante el ciclo de vida: Una vez que el esfuerzo total y la duración del proyecto ha sido estimados, han de ser distribuidos a lo largo del ciclo de vida.

- Reflejar el esfuerzo en el calendario: El paso final del proceso de *Arifoglu* es convertir el número de días de trabajo requeridos para completar el proyecto en número de días transcurridos en el calendario. *Arifoglu* es partidario del uso del modelo de *Esterling*, para realizar esta conversión. (Esterling, 1980) propone un modelo para estimar el porcentaje de jornada laboral utilizado en la realización directa una determinada tarea. Este paso realiza la predicción de la duración del proyecto usando el modelo *COCOMO* u otro similar. La duración predicha por *COCOMO* es ya un tiempo estimado de calendario (incluyendo fines de semana, y tiempo medio de vacaciones y bajas por enfermedad).

6. Proceso de *Humphrey* (*PROBE*).

(Humphrey & Watts, 1995) Proponen un proceso de estimación basada en *proxys* (*Proxy-Based Estimation PROBE*) como parte de su proceso de *Software* personal. El método *PROBE* defiende el uso de medidas seleccionadas personalmente y modelos de regresión para estimar el tamaño de un producto *software*. Un *proxy* es una medida de tamaño que puede ser utilizada para estimar la longitud de un proyecto, medida en líneas de código. Por ejemplo *PROBE* utiliza una cuenta de objetos como

un proxy de medida de tamaño. La estimación de líneas de código se utiliza para predecir el esfuerzo individual para desarrollar el producto. Los límites superior e inferior del intervalo de predicción deben ser también estimados.

Se utilizan modelos de regresión simples para estimar las líneas de código desde las medidas de tamaño basadas en proxys y las horas de trabajo desde las líneas de código. Se realiza una vuelta atrás desde cada estimación, comparando el valor estimado con el real. La vuelta atrás se introduce para mejorar el rendimiento de la estimación individual. La filosofía del Proceso Personal del *software* es mejorar las habilidades de los individuos lo cual dificulta en cierto modo la aplicabilidad de *PROBE* a equipos de trabajo.

7. *Wide band Delphi.*

Es posible que la precisión de un método de estimación pueda ser mejorando mediante la opinión y la estimación de un grupo de personas. Para (Boehm., 1981) se describe la técnica *Wide band Delphi* para combinar las opiniones de los expertos y realizar una estimación de tamaño.

A cada experto se le proporciona una especificación y un formulario de estimación. En una reunión los expertos discuten los asuntos de la estimación. Cada experto rellena su formulario de forma anónima. En esta ronda, se proporciona a los expertos un sumario de las estimaciones

realizadas, posteriormente se mantiene otra reunión para discutir las estimaciones de la ronda anterior. Se celebrarán más rondas hasta que las estimaciones converjan satisfactoriamente.

Wide band Delphi es similar al *Delphi* original, desarrollado por *RAND Corporation* (Helmer, 1967) no obstante, antes de hacer cada ronda de estimaciones, los expertos discuten sobre las estimaciones anteriores. En la técnica original no había interacción entre los expertos.

8. *Estimeeting.*

(Trafalis, 1999) Describe otro proceso para la estimación en grupo, *Estimeeting*, el cual ha sido usado por *AT&T Bell Laboratories.* Se mantiene una reunión conocida como *Estimeeting* para producir una estimación del esfuerzo para desarrollar una caracterización del sistema.

Antes de la reunión, un grupo de estimadores revisan los requerimientos y una caracterización de alto nivel del sistema. Estos estimadores no son parte del equipo de desarrollo, pero son expertos en subsistemas y serán cambiados cuando la característica en cuestión del sistema sea establecida. El equipo de trabajo que la establece, presenta los requerimientos y el diseño de los atributos de estimación. A la presentación le sigue un periodo de preguntas y respuestas. Entonces los estimadores realizan estimaciones para los subsistemas en los que son expertos. Los estimadores consultan entre ellos y con el equipo que establece la característica. El equipo que determina la característica es

responsable de sumarizar los resultados de la reunión y calcular la estimación total.

A diferencia de *Wideband Delphi* que determinaba la estimación final por consenso, los resultados de *Estimeeting* son la conjunción de todas las estimaciones individuales.

Además permite estimaciones en mayor detalle, pero requiere un diseño de alto nivel, en el que se conozcan los requerimientos de los subsistemas. *Wideband Delphi* tiene la ventaja del consenso en cada estimación, pero el detalle de la estimación es menor, y cada estimador duplica el trabajo de los otros.

I.1.5. ESTADÍSTICAS DE LOS PROYECTOS DE SOFTWARE

Mckinsey & Company, junto con la universidad de *Oxford*, realizaron un estudio publicado en Octubre de 2012, enfocado en grandes proyectos de tecnología de la información (a partir de este momento denominado *TI*). Estos proyectos de *TI* tenían presupuestos iniciales de más de 15 millones de dólares.

La investigación fue desarrollada por *Michael Bloch, Sven Blumberg, y Jürgen Laartz.* De acuerdo a esta investigación, de los 5,400 proyectos de *TI* consultados, 45% han excedido su presupuesto, 7% han excedido su cronograma y 56% entregan menos valor que el predicho. También el

17% de los proyectos de *TI* van tan mal, que pueden poner en peligro la existencia misma de la empresa.

Después de comparar presupuestos, horarios y beneficios previstos de rendimiento con los costes reales y los resultados, se encontró que los proyectos de *TI*, en total, tenían un sobrecosto de $ 66 mil millones, más que el *PIB* de Luxemburgo. Asimismo, se encontró que cuanto más tiempo un proyecto este programado para durar, lo más probable es que se ejecutará en el tiempo y el presupuesto, además con cada año adicional gastado en el proyecto el aumento en los costes es aproximadamente un 15%.

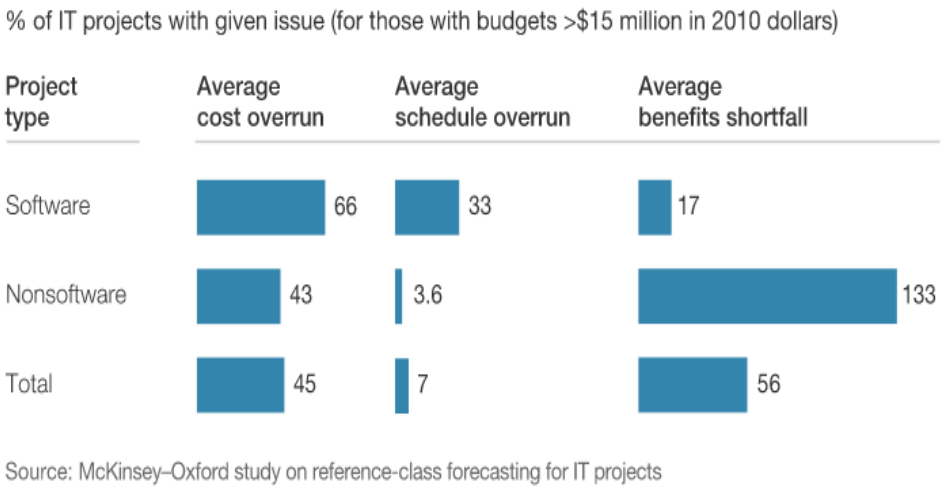


Imagen 6: Estadísticas de proyectos de TI. Fuente *McKinsey & Company*

El estudio también da porcentajes según tipo de proyecto, sean proyectos de *software* o no.

Estos eventos de alto impacto (*black swans*, cisnes negros[1]) ocurren con mucha más frecuencia de lo esperado bajo una distribución normal. Los grandes proyectos de *TI* que se convierten en cisnes negros se definen como los que tienen excesos de presupuesto de más de 200 % (y hasta 400 % en el extremo final).

Tales excesos igualan o superan a las experimentadas por los cisnes negros de los proyectos de construcción complejos, como túneles y puentes. Un *retail* comenzó un esfuerzo de $ 1.4 mil millones para modernizar sus sistemas de *TI*, pero el proyecto fue abandonado. A medida que la empresa cayó detrás de sus competidores, se inició otro proyecto, un nuevo sistema de gestión de la cadena de suministro, por una suma de $ 600 millones. Cuando ese esfuerzo también falló, la empresa tuvo que declararse en quiebra. (Bloch, Blumberg, & Laartz, 2012, pág. 1)

I.2. PLANTEAMIENTO DEL PROBLEMA

I.2.1. DESCRIPCIÓN DE LA PROBLEMÁTICA

La descripción de la realidad problemática del trabajo de investigación presenta tres aristas importantes, procesos, personas y tecnología. Explicadas cada una a continuación:

[1]La teoría del cisne negro o teoría de los eventos del cisne negro es una metáfora que encierra el concepto de que cuando un evento es una sorpresa y tiene un gran impacto, después del hecho, este evento sorpresivo es racionalizado por retrospección

- **Procesos.**

1. No existe un modelo o metodología de estimación universal o una fórmula que pueda ser usada para todas las organizaciones. El hecho es que se pueden definir unos principios generales, pero cada interpretación es particular y diferente del resto. Cada organización tiene sus propios recursos, procedimientos e historia (*APO*); y es necesario ajustar los procesos de estimación a esos parámetros únicos. Además, a medida que estos parámetros cambian, deben de cambiar los procesos de estimación.

2. La utilidad de una estimación también dependerá del método de desarrollo que se adopta (*waterfall - agile*) y en la etapa en la que nos encontremos. Al comienzo de un proyecto, normalmente sólo se necesitan estimaciones de coste y duración aproximadas. A medida que el proyecto madura las estimaciones que se necesitan serán más exactas. Con lo que una estimación útil al comienzo del proyecto puede no serlo más tarde. No obstante el esfuerzo y el presupuesto del proyecto ya fueron definidos en la etapa inicial.

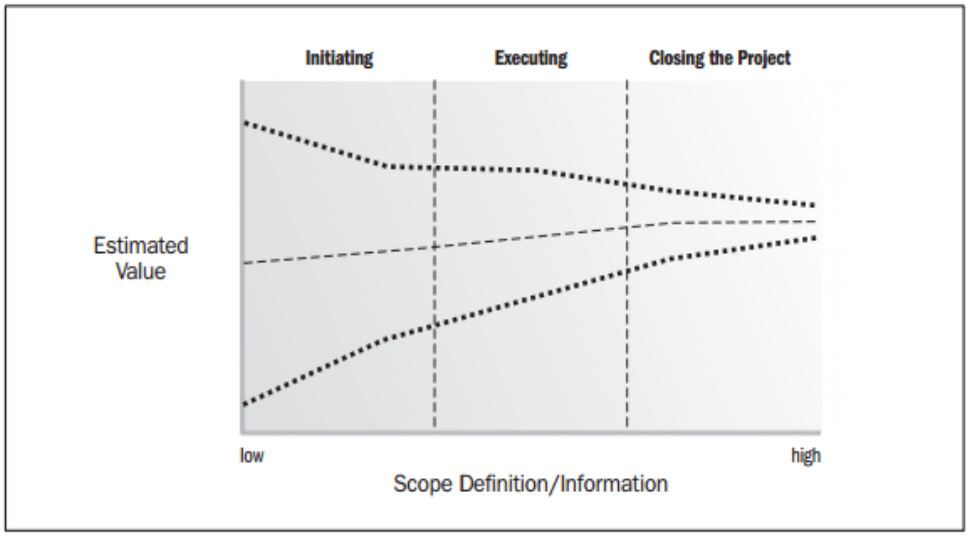


Imagen 7: Evolución de alcance y los rangos de estimación del proyecto. Fuente: *Practice Standard for Project Estimating - PMI*

3. Las estimaciones claras, completas y precisas son difíciles de formular, especialmente al inicio del proyecto. Los cambios, adaptaciones y ampliaciones son más la regla que la excepción, como consecuencia de ello, deben adaptarse también las planificaciones y los objetivos. En la práctica, el número e impacto de las solicitudes de cambio a menudo se incrementa al final del proyecto. Esto puede deberse a cualquier combinación de factores, incluyendo proyectos con un alcance poco delimitado, falta de apropiación de requerimientos por parte de los *stakeholders* del proyecto, pobre realización del análisis de negocio, cambio de prioridades de negocio, reorganización del negocio, cambios regulatorios o requerimientos cambiantes del negocio.

4. Las características del *software* y de su desarrollo hacen difícil la estimación. Por ejemplo, el nivel de abstracción, la complejidad, el grado de medición del producto y del proceso, los aspectos innovadores, entre

otros. Los proyectos de innovación, proyectos de investigación y mercadeo tienden a tener mayor grado de incertidumbre en los requerimientos mientras que los proyectos de contabilidad y finanzas tienden a tener un nivel relativamente menor de incertidumbre en los requerimientos.

- **Personas.**

1. Hay muchos *stakeholders* implicados en los proyectos que precisan de estimaciones. La alta dirección de la empresa y/o el patrocinador necesitan las estimaciones para tomar decisiones de negocio, sobre la viabilidad del proyecto y su continuidad a lo largo del desarrollo (*business need – business case*). La dirección del proyecto necesita las estimaciones para hacer sugerencias a la alta dirección, para obtener los resultados necesarios para el desarrollo del proyecto, y para hacer una planificación detallada y controlar el proyecto (*initiating processes - planning processes*). Cada recurso del proyecto también necesita estimaciones para planificar y controlar su propio trabajo (*executing processes*).

2. La estimación, a menudo, se hace superficialmente, sin apreciar el esfuerzo requerido para hacer un trabajo. Además, también se suele dar el caso de que la estimación sea necesaria antes de obtener las especificaciones de requisitos del sistema (elicitación). Por esta razón, una situación típica es que se presiona a los estimadores para que se apresuren en escribir una estimación anticipada del sistema que no comprenden aún.

El *PMI* recomienda que un proyecto en su fase de inicio puede tener el rango de −25% a +75% (*ROM rough order of magnitude, project charter*), **sí y solo sí el proyecto no es producto de un acuerdo/contrato.** En una etapa posterior del proyecto, conforme se va contando con más información, el rango de exactitud de las estimaciones puede reducirse a -5% a +10% (*planning processes*)

3. Un estimador puede no tener mucha experiencia en estimar el desarrollo, especialmente de proyectos largos. ¿Cuántos proyectos largos puede alguien dirigir por ejemplo en 10 años?

4. Existe una tendencia aparente de los desarrolladores hacia la subestimación o al relleno (*padding)*. Un estimador suele elegir una porción de *software* que debería tomar, para luego extrapolarlo al resto del sistema, normalmente se ignoran los aspectos no lineales del desarrollo de *software*, por ejemplo, la coordinación y la gestión.

5. Los estimadores (líder técnico del proyecto), estiman el tiempo que le llevaría ejecutar personalmente una tarea, ignorando el hecho de que, a menudo, una parte del trabajo la realiza personal menos experimentado, con un índice de productividad menor.

6. Existen malas interpretaciones en las relaciones lineales entre la capacidad requerida por unidad de tiempo y el tiempo disponible. Esto significa que el *software* desarrollado por 25 personas en dos años podrá ser llevado a cabo por 50 personas en un año (mal empleo de la técnica de compresión

del cronograma (*crashing - fast tracking)*). Esta interpretación es errónea. Como se sabe una mujer puede dar a luz un niño a lo largo de 9 meses, pero 9 mujeres no dan a luz un niño en un mes. Añadir personal a un proyecto retrasado no tiene por qué disminuir el retraso. *Ley de retorno decreciente.*

7. El estimador tiende a reducir en algún grado las estimaciones para hacer más aceptable la oferta. Asumiendo un riesgo innecesario que va en contra del proyecto.

8. Influyen un gran número de factores en el esfuerzo y duración de un desarrollo de *software*. Estos factores se llaman "*drivers de coste*" o disparadores de coste. Ejemplos de estos disparadores son el tamaño y complejidad del *software*, el compromiso y participación de los *stakeholders*, la experiencia del equipo de desarrollo. En general estos disparadores de coste son difíciles de determinar.

- **Tecnologías.**

1. La rapidez con la que cambia la tecnología de la información (*software* base) y las metodologías de desarrollo de *software* son problemas para la estabilización del proceso de estimación. Por ejemplo, son difíciles de predecir la influencia de nuevos bancos de pruebas, lenguajes de cuarta y quinta generación, estrategias de prototipado, y de técnicas y herramientas novedosas en general.

2. Las diferentes arquitecturas de desarrollo implementadas en las aplicaciones de una organización.

3. La no estandarización de un *framework* de desarrollo para todas las aplicaciones de la organización. (*white paper*).

4. Los tipos de sistemas, ya que no es lo mismo estimar un aplicación transaccional (*OLTP*), que un proyecto de *BI* (*OLAP*), en el primer caso se emplean o crean objetos nuevos y tienen procesos atomizados, para el segundo caso hay que realizar limpieza de datos, analizar datos no estructurados y estructurados, n tipos de fuentes entre otras variables.

I.2.2. DESCRIPCIÓN DEL PROBLEMA

Después de analizar la realidad problemática podemos definir las siguientes interrogantes.

- ¿Existen múltiples técnicas de estimación de *software* pero no existe una metodología integral que permita la estimación de proyectos de *software,* desde la elicitación de requisitos hasta la planificación del proyecto, y que esté basada en los cuerpos de conocimiento del *PMBOK* y el *BABOK*?

- ¿Cuántas organizaciones tienen una *BD* de conocimiento de lecciones aprendidas?

- ¿Cuántas organizaciones emplean la *BD* de conocimiento de lecciones aprendidas de la organización, retroalimentando los nuevos proyectos

con el descubrimiento de conocimiento de datos (*Knowledge Discovery From Databases* a partir de este momento denominado *KDD*) de las casuísticas estructuradas presentadas en los proyectos *post morten*?

- ¿Qué técnicas de estimación funcional, son independientes de la tecnología y que permitan medir lo que el usuario pide y lo que el usuario recibe?

- ¿Qué técnicas de elicitación de requisitos se pueden emplear de manera que no solamente estén enfocados en las necesidades, (meta modelo diseñado para facilitar la elicitación y representación del lenguaje usado en la aplicación) sino también en la estimación de cada requisito?

I.2.3. FORMULACIÓN DEL PROBLEMA

I.2.3.1. PROBLEMA GENERAL

¿De qué manera el diseño y puesta en marcha de una metodología integral que permita la estimación funcional de proyectos de *software*, desde la elicitación de requisitos (*BABOK*), hasta la planificación del proyecto (*PMBOK*) complementada con las técnicas de estimación funcional (*UFP*, *SMC/RICEF*) y el léxico extendido del lenguaje (*LEL*), que sea retroalimentada por el descubrimiento de conocimiento de datos (*KDD*) de

las casuísticas estructuradas presentadas en los proyectos post morten, puede mejorar la gestión de proyectos de *software*?

I.2.3.1. PROBLEMA ESPECÍFICO

- ¿En qué medida la utilización de la estimación funcional sobre una metodología integral basada en el descubrimiento de conocimiento de datos, contribuye en la eficiencia de la gestión de proyectos de software en las organizaciones bajo estudio?

- ¿En qué medida la utilización de la estimación funcional sobre una metodología integral basada en el descubrimiento de conocimiento de datos, contribuye en la eficacia de la gestión de proyectos de software en las organizaciones bajo estudio?

- ¿En qué medida la utilización de la estimación funcional sobre una metodología integral basada en el descubrimiento de conocimiento de datos, contribuye en la efectividad de la gestión de proyectos de software en las organizaciones bajo estudio?

I.3. OBJETIVOS

I.3.1. OBJETIVO GENERAL

Establecer el grado de mejora de la gestión de proyectos de *software* al emplear una metodología integral que permita la estimación de proyectos de *software*, desde la elicitación de requisitos (*BABOK*), hasta la planificación del proyecto (*PMBOK*) complementada con las técnicas de estimación funcional (*UFP*,

SMC/RICEF) y el léxico extendido del lenguaje (*LEL*), que sea retroalimentada por el descubrimiento de conocimiento de datos (*KDD*) de las casuísticas estructuradas presentadas en los proyectos *post morten*.

I.3.2. OBJETIVO ESPECÍFICOS

- Establecer el grado de eficiencia en la gestión de proyectos de software, al utilizar la estimación funcional sobre una metodología integral de estimación, basada en el descubrimiento de conocimiento de datos.

- Establecer el grado de eficacia en la gestión de proyectos de software, al utilizar la estimación funcional sobre una metodología integral de estimación, basada en el descubrimiento de conocimiento de datos.

- Establecer el grado de efectividad en la gestión de proyectos de software, al utilizar la estimación funcional sobre una metodología integral de estimación, basada en el descubrimiento de conocimiento de datos.

I.4. JUSTIFICACIÓN E IMPORTANCIA

I.4.1. JUSTIFICACIÓN

La investigación servirá para, mejorar la estimación de los proyectos de *software*. Al reducir las estimaciones superficiales que van en contra del proyecto (costo, tiempo, calidad y productividad)

Los beneficios de la investigación son direccionados a la comunidad de profesionales vinculados en la gestión de proyectos de *software*, ya que se les proporcionará una metodología integral de estimación.

La información obtenida estará relacionada al grado de mejora de la estimación de los proyectos, al tener una metodología integrada con la retroalimentación de la *BD* de conocimiento de los mismos.

I.4.2. IMPORTANCIA

Los resultados de la investigación podrán beneficiar a las empresas estatales y/o privadas y a los profesionales vinculados en la estimación de proyectos de *software*, dejando de lado las estimaciones superficiales que van en contra del proyecto y más cuando estas se tienen que realizar al inicio del mismo.

I.5. ALCANCES Y LIMITACIONES

I.5.1. ALCANCE

El presente trabajo tiene como alcance el desarrollo de una metodología de procesos bajo el enfoque de administración basada en planes (*waterfall*), emplea algunas tareas y procesos de los cuerpos de conocimiento del *BABOK* y el *PMBOK*, la técnica de puntos de función *UFP* para proyectos mayores a 100 *PF*, la técnica *SMC/RICEF* para proyectos menores o iguales a 100 *PF* y el léxico extendido del lenguaje (*LEL*) para la elicitación de los requisitos. Toda esta metodología es retroalimentada mediante el *KDD* (*Knowledge Discovery From Databases*) empleando los algoritmos de *Data Mining* como *Cluster* y *Neural Network*.

Sólo comprenderá el dimensionamiento funcional en puntos función de cada proyecto, la cantidad de horas involucradas de cada uno de ellos y los costos relacionados, no contempla los procesos de evaluación de productividad, calidad y riesgos.

Además comparará el comportamiento del proyecto en las etapas de viabilidad (línea base viabilidad), análisis (línea base análisis), diseño (línea base diseño), planificación (línea base planificación) contra el cierre del mismo, evaluando sus variaciones en cada etapa, en función a los objetivos previamente definidos.

I.5.2. LIMITACIÓN

El presente trabajo tiene como principal limitación el entorno de la muestra de datos para el aprendizaje de los algoritmos de *Cluster* y *Neural Network*, ya que se limitará exclusivamente a los proyectos realizados en Edelnor y Stefanini. Debido a que son pocas las empresas en nuestro país que emplean las técnicas de UFP y SMC/RICEF.

I.5.2.1. DELIMITACION ESPACIAL

Esta investigación recopilará y analizará la información referente al problema de la estimación de los proyectos de *software* en Edelnor y Stefanini.

I.5.2.2. DELIMITACION TEMPORAL

El objeto de la investigación tomará como punto de partida las estimaciones realizadas en:

- Edelnor: desde el año 2011 hasta 2016

- Stefanini: desde el 2014 hasta el 2016

II. MARCO TEÓRICO

II.1. TEORÍAS GENERALES RELACIONADAS CON EL TEMA

II.1.1. ESTIMACIÓN DEL TAMAÑO DEL SOFTWARE

Para (Chang, 2001, págs. 239-277), esta actividad se refiere a la necesidad de conocer a ciencia cierta qué tan grande va a ser el *software* que se va a construir y lograr conocer de manera tangible el costo de desarrollar un sistema basándose en una medición acertada acerca del tamaño del *software*.

La estimación del tamaño del *software* se puede realizar en diferentes etapas del proyecto se recomienda mínimo 3 etapas:

- Elicitación de los requisitos.
- Análisis funcional de los requisitos.
- Análisis técnico de los requisitos.

Dependiendo del período en que ésta se lleve a cabo, es posible determinar su correspondencia con el tamaño real del *software*. Por ejemplo, si la estimación se realiza al final del proyecto se puede realizar una estimación, por así decirlo 100% acertada, debido a que para este momento ya se conoce la duración total de éste (*post mortem*), además de la cantidad de código escrito. Sin embargo, si la estimación se realiza en etapas tempranas del proyecto se podría decir que el resultado estaría bastante alejado de la realidad.

Lo realmente importante de la estimación no es necesariamente que ésta sea 100% confiable, sino el hecho de que su realización contribuya en la determinación del costo total del proyecto, por lo cual, se recomienda que durante el desarrollo de éste se realicen estimaciones y se corrijan las anteriores con la información que se vaya recolectando, lo que a largo plazo, ayuda a que las estimaciones que se hagan sobre proyectos futuros sean cada vez más acertadas.

Para la realización de esta actividad existen diversos métodos y metodologías, pero las metodologías más destacadas para la estimación del tamaño del *software* son el conteo de líneas de código del programa producido y el conteo de puntos de función. Sin embargo, en este tipo de estimaciones no se tienen en cuenta los documentos que se deben generar cuando se está construyendo *software*. Dichos documentos también requieren tiempo y recursos, que incrementan el tamaño del *software* en desarrollo.

II.1.1.1. METODOLOGÍAS DE ESTIMACIÓN DEL TAMAÑO DEL *SOFTWARE*

A continuación se presenta una descripción de cada una de las metodologías de estimación del tamaño, consideradas como las más importantes y más usadas por la industria. Como fue mencionado anteriormente existen básicamente dos aproximaciones a esta estimación: el conteo de líneas de código y el conteo de puntos de función. A continuación describiremos dichas aproximaciones (Chang, 2001, págs. 239-277).

II.1.1.1.1. ESTIMACIÓN BASADA EN LÍNEAS DE CÓDIGO.

Esta estimación se podría catalogar como de tipo tardío (*Post Mortem*), ya que el número total de líneas de código sólo se puede conocer cuando el producto esté terminado, aunque la tarea no es tan sencilla como contar la longitud de cada archivo; se debe acordar un formato, en donde se especifique qué es lo que se va a contar y qué no. Por ejemplo, los comentarios escritos en el código no deberían ser contados, por lo cual sólo se debe contar, lo que se especifique a ser contado.

Dentro de esta categoría existen varias metodologías las cuales usan las líneas de código como base para la realización de su estimación. A continuación se explican algunas de estas metodologías.

- **ESTIMACIÓN POR CONTEO DE BLOQUES**

Este enfoque se basa en estimar el número de funciones esperadas que tendrá el sistema. Se puede ver como un enfoque de estimación temprana debido a que estima el número de funciones esperadas. Por tanto, se cuenta con poca información acerca del proyecto con lo que las estimaciones no podrían ser muy exactas. De esta manera, a medida que avanza el proyecto es deseable que las estimaciones fueran más coherentes con la realidad.

Es posible que el método pueda ser complementado con funciones estadísticas para encontrar una estimación más precisa. Con este fin es usada la desviación estándar, obtenida a partir de la información de proyectos pasados ya realizados, lo cual mejora en gran parte las estimaciones para la organización.

A continuación se enumeran los pasos empleados en el uso de este modelo:

a. Estimar el número de bloques, o componentes de *software* esperados.

b. Multiplicar el número de bloques por el tamaño esperado de cada tipo de bloque.

c. Calcular la desviación estándar para dicho proyecto.

d. Aplicar el método repetidamente para los diferentes niveles de detalle, y así obtener una estimación más precisa.

- **ESTIMACIÓN DEL TAMAÑO ESTADÍSTICA**

Este método se basa en la estimación del tamaño a partir de la utilización de cálculos estadísticos y dividiendo el sistema en componentes para cada uno de los componentes que integran el sistema, posibilitando la estimación del sistema completo tomando como base cada uno de sus componentes por separado. Asimismo, este método se encarga de disminuir la incertidumbre acerca de las estimaciones de los componentes individuales, lo cual posibilita

contar con una estimación mucho más segura del sistema completo. Con este fin, el método se basa en la estimación por analogía, en la cual se compara el proyecto actual con otros anteriores ya realizados, evidenciando la necesidad de mantener una base de datos con la información acerca de todos estos proyectos anteriores que servirán para la estimación del proyecto, a continuación se listan los pasos para estimar el tamaño del *software* con este método:

a. Determinar las funciones que compondrán el nuevo sistema.

b. Buscar información acerca del tamaño de funciones similares ya desarrolladas.

c. Identificar las diferencias entre las funciones similares y las nuevas.

d. Para cada componente o función a estimar, se deben estimar tres parámetros, el menor, medio y máximo tamaño de cada uno de los componentes o funciones.

e. Calcular la media estadística y desviación estándar de cada uno de los números obtenidos en el numeral anterior.

f. Tabular cada uno de estos datos obtenidos.

g. Calcular la media total del proyecto, y la desviación estándar del proyecto.

- **ESTIMACIÓN POR LÓGICA DIFUSA**

Este método se basa en dividir el proyecto en categorías de tamaño y dependiendo de la cantidad de líneas de código producidas en cada una clasificarlas en grande, mediano y pequeño. Para realizar esta categorización se requiere tener información de proyectos anteriores para generar los grupos antes descritos.

Por consiguiente para realizar la estimación del nuevo proyecto se debe juzgar en qué categoría quedaría éste, lo cual daría un rango de líneas de código que el nuevo proyecto podría producir.

Un problema que presenta este método, es que el cambio tecnológico trae como consecuencia que la magnitud en líneas de código de un proyecto varié, lo cual podría hacer que los grupos ya anteriormente definidos necesariamente tengan que cambiar.

II.1.1.1.2. ESTIMACIÓN BASADA EN ANÁLISIS DE PUNTOS DE FUNCIÓN FPA - IFPUG 4.3.1

Para Longstreet(Longstreet, 2004), esta técnica se diferencia a los basados en líneas de código en que, no se basa en la longitud de programa sino en la funcionalidad que presta, lo cual hace a este método independiente del lenguaje.

El análisis de punto función es una técnica que, mediante la descomposición de un sistema en componentes más pequeños, permite que éstos puedan ser mejor comprendidos y analizados en forma individual.

El análisis de punto función se basa en la teoría de que las funciones de una aplicación son la mejor medida del tamaño de un sistema. El punto función mide el *software* mediante la cuantificación de la funcionalidad que el sistema le brinda al usuario basado fundamentalmente en el diseño lógico. Es independiente del lenguaje de computación, de la metodología de desarrollo, de la tecnología utilizada y de la capacidad del equipo de trabajo para desarrollar la aplicación.

El análisis del punto función es una técnica estándar de medición de desarrollo de *software* desde el punto de vista del usuario. Su objetivo es medir el *software* basándose en la cuantificación de la funcionalidad brindada al usuario partiendo fundamentalmente de diseños lógicos. La cuenta de punto

función para proyectos de desarrollo mide las funcionalidades que se le proporcionan al usuario conjuntamente con la primera instalación del *software* producido cuando el proyecto es terminado.

El *IFPUG International Function Points Users Groups* o, en español, el grupo internacional de usuarios de puntos función. El grupo fue creado en 1986 para mejorar la técnica previamente definido por *Albrecht FPA*.

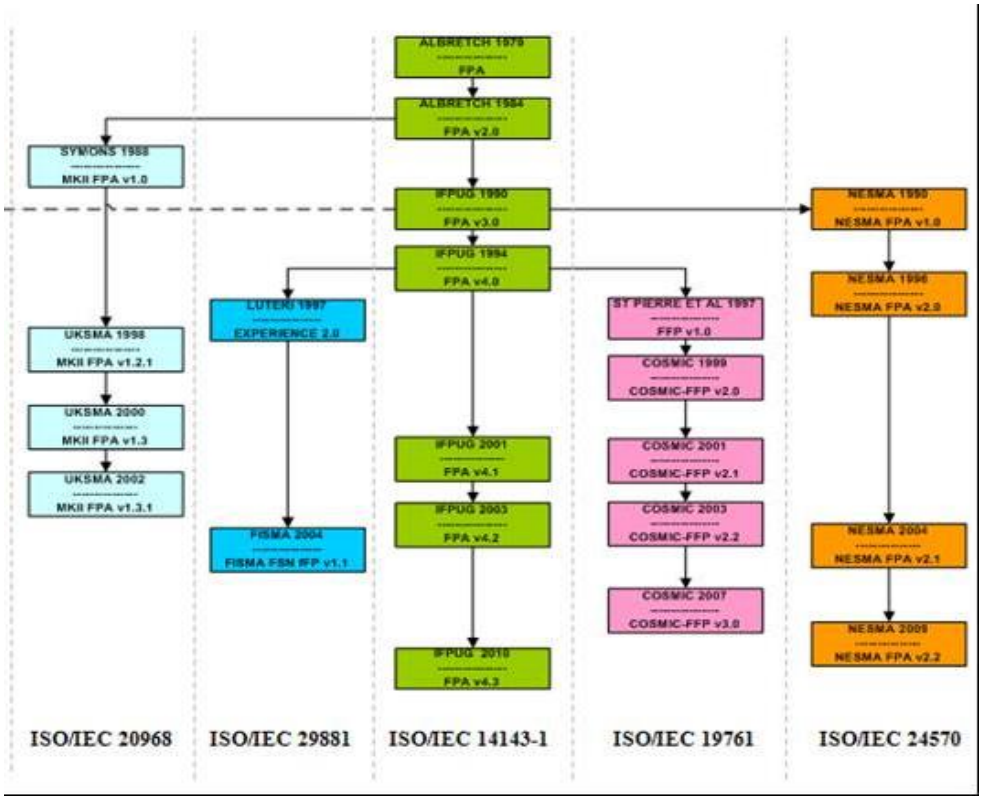


Imagen 8. Evolución de la técnica de puntos función y las ISO/IEC correspondientes
Fuente: (Gonzáles Mateo, 2012)

IFPUG mantiene un manual donde describe el método de conteo en puntos función es el *Counting Practices Manual* la versión actual es la 4.3.1. Siendo la última versión del 2010.

Esta técnica se basa en dividir la funcionalidad del sistema a medir en dos tipos de funciones:

- **Funciones transaccionales.** Estas funciones también llamadas procesos elementales intentan modelar las necesidades de procesos del

usuario, entendiendo por usuario a cualquier persona o cosa que interactúa con el sistema que estamos midiendo. Algunos ejemplos de funciones transaccionales: alta de empleado, listado de empleados, informe mensual de empleados.

- **Funciones de Datos.** Estas funciones tratan de modelar las necesidades de almacenamiento de información que tiene el usuario (usuario definido como en el punto anterior). Las funciones de datos también se denominan grupos lógicos y son una generalización del concepto de entidad pero debemos tener cuidado porque aunque pueden coincidir no siempre es así.

El valor de una medición en puntos función de un sistema es la suma del valor de los puntos función de sus funciones transaccionales y de sus funciones de datos.

El valor en puntos función de cada una de las funciones se determina a través de sus componentes.

Para las **funciones transaccionales** sus componentes son:

- Los atributos que envía o recibe la propia función. Algunos ejemplos, los campos de un formulario, el resultado de una consulta.

- Los grupos lógicos o funciones de datos que utiliza. Algunos ejemplos, la entidad de empleado al dar de alta un elemento nuevo.

Para las **funciones de datos** sus componentes son:

- Los atributos que almacenan. Algunos ejemplos, código del empleado, nombre del empleado, departamento del empleado.

- Los grupos de atributos que se pueden identificar siguiendo unas determinadas reglas definidas en el método. Algunos ejemplos, empleado, beneficiarios del empleado.

Estos valores permiten establecer la complejidad de cada función según unas tablas prefijadas en el método. Los valores de la complejidad son baja, media o alta.

Según el tipo de función transaccional o de datos la complejidad se convierte en un valor en puntos función a través de unas tablas definidas en el método.

Aplicando todo ello obtenemos una valoración de nuestro proyecto en puntos función que se denominan no ajustados.

Hasta aquí sería el estándar *ISO* del método de *IFPUG* aunque el método incluye una serie de pasos más que últimamente están cayendo en desuso y es el factor de ajuste.

El factor de ajuste es un valor que se calcula para cada sistema/aplicación que estemos midiendo evaluando para ello 14 características del sistema. Los valores de cada característica van de 0 a 5 y todos ellos se suman y se les aplica una fórmula que pueden hacer variar la valoración de puntos función no ajustados en un -35% o +35%.

Las 14 características generales del sistema son:

1. Comunicaciones de datos
2. Procesamiento de datos distribuido
3. Rendimiento
4. Configuración altamente utilizada
5. Tasa de transacciones
6. Entrada de datos *On-line*
7. Eficiencia del usuario final
8. Actualización *On-line*

9. Complejidad de procesamiento
10. Reusabilidad
11. Facilidad de instalación
12. Facilidad de operación
13. Múltiples localizaciones
14. Facilidad de cambio

Con ello tendríamos los puntos de función ajustados para nuestro proyecto.

A lo largo de esta Tesis, **NO INCLUIREMOS** estos factores de ajuste como parte del propio método, destacando que su uso es puramente opcional, pudiéndose realizar mediciones mediante esta métrica sin aplicar dichos factores, lo que nos generaría un resultado en *UFP* (*Unadjusted Function Points*) en lugar de los hasta *FPA* (*Adjusted Function Points*).

VENTAJAS DE LA TÉCNICA DE PUNTOS DE FUNCIÓN:

- Es independiente de la tecnología.
- Se puede usar en cualquier fase del ciclo de vida de un proyecto *software*.
- Es un método que está muy bien documentado.
- Soporta la elaboración de una planificación realista.
- Mide lo que el usuario pide y lo que el usuario recibe.
- Soporta la comunicación entre la administración, los usuarios y proveedores.
- Cumple con la norma *ISO 14143*.

- Existe una gran cantidad de datos históricos.

DESVENTAJAS DE LA TÉCNICA DE PUNTOS DE FUNCIÓN:

- Sin datos históricos es difícil mejorar las habilidades de estimación.
- Los puntos función no reflejan diferencias entre lenguajes diseño o estilos.
- El factor de ajuste calculado a partir de las características generales del sistema resulta de dudosa utilidad.
- Carece de precisión cuando se realiza en proyectos pequeños con menos de 100 puntos función.

ISACA (*Information Systems Audit and Control Association*) publicó un artículo de investigación ("*Understanding Software Metric Use*", *ISACA Journal, volume 1, 2015.)* donde se encuestaron a 126 gerentes, desarrolladores, y coordinadores en métricas en los EE.UU. Para obtener conocimientos sobre las técnicas de métricas de mayor uso.

De allí se obtiene que puntos de función se encuentra como la métrica número 1 con el 39 % del total, seguido de líneas de código con un 21%.

Utilización de técnicas de métricas:

- Puntos de función - 39 %.
- Líneas de código - 21 %.
- Número de defectos - 18 %.
- Horas estimadas - 12 %.
- Tamaño estimado - 6 %.
- Otros complejidad - 2 %.

- Complejidad ciclomática - 1 %.
- Análisis de complejidad - 1 %.

II.1.2. TIPOS DE SISTEMAS DE INFORMACIÓN

Según Kendall k. y Kendall J, (Kendall & Kendall, 2005), se pueden dividir en 4 grandes grupos

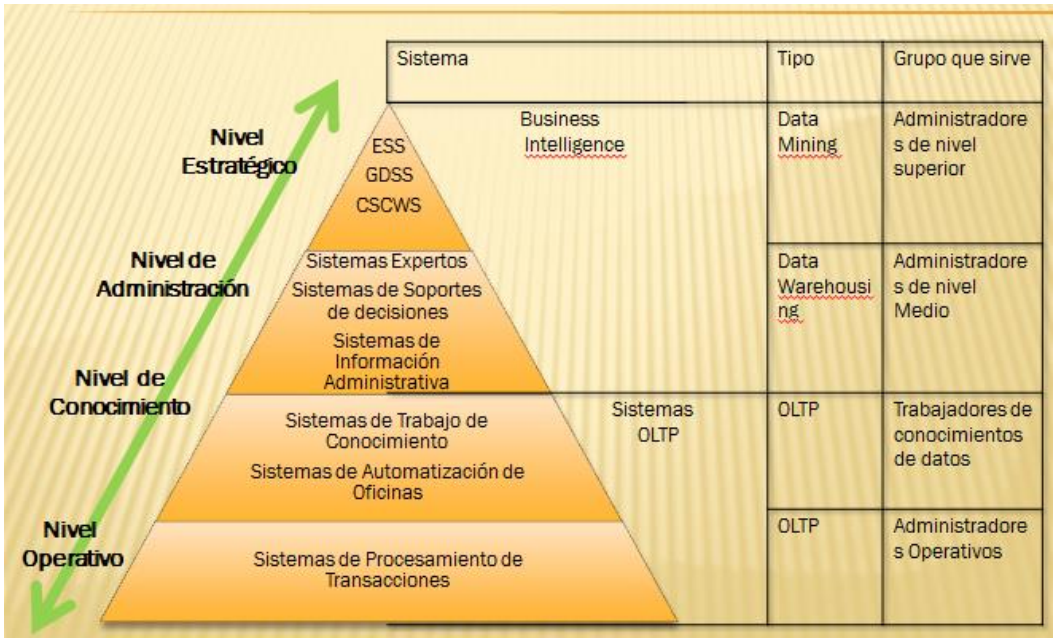


Imagen 9. Tipos de sistemas Fuente: (Kendall & Kendall, 2005)

II.1.2.1. SISTEMAS DE PROCESAMIENTO DE TRANSACCIONES.

- **Sistemas de procesamiento de transacciones (*TPS Transaction Prosesing Systems*)** funcionan al nivel operativo de una organización y procesan las transacciones rutinarias de las organizaciones (nóminas, contabilidad, ventas, recursos humanos, etc.) (Kendall & Kendall, 2005, pág. 2).

  Arquitectura recomendada para este tipo de sistemas

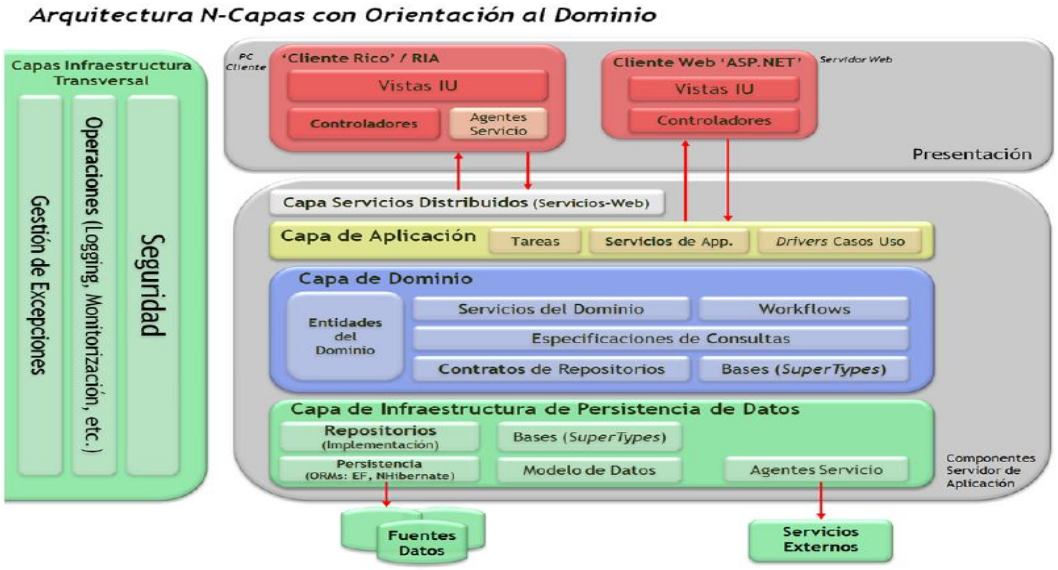


Imagen 10: Arquitectura N capas. Fuente (De la Torre, Zorrilla, Ramos, & Calvarro, 2010, pág. 56)

II.1.2.2. SISTEMAS DE GESTIÓN DE INFORMACIÓN.

- **Sistemas de Automatización de Oficinas (*OAS Office Automation Systems*).** Son los llamados sistemas de ofimática, no crean un nuevo conocimiento sino que usan la información para analizarla y transformar datos, o para manejarla en alguna forma y luego compartirla o diseminarla formalmente por toda la organización y algunas veces más allá de ella (procesamiento de textos, hojas de cálculo, gráficos, correo electrónico, videoconferencias, etc.) (Kendall & Kendall, 2005, pág. 3).

- **Sistemas de Trabajo de Conocimiento (*KWS Knowledge Work Systems*)** apoyan el trabajo al nivel del conocimiento, dan soporte a los trabajadores profesionales, tales como científicos, ingenieros y doctores, ayudándoles a crear nuevos conocimientos que contribuyan a mejorar la organización. (Kendall & Kendall, 2005, pág. 3)

  La arquitectura recomendada es similar para sistemas de procesamiento de transacciones.

II.1.2.3. SISTEMAS DE GESTIÓN TÁCTICO

Según Kendall k. y Kendall J, (Kendall & Kendall, 2005), se pueden dividir en tres grupos.

- **Sistemas de Información Gerencial (*MIS Management Information Systems*).** Producen información que es usada para la toma de decisiones, basándose en los sistemas de procesamiento de transacciones. En otras palabras, dan soporte a un espectro más amplio de tareas organizacionales que los sistemas de procesamiento de transacciones, incluyendo el análisis de decisiones y la toma de decisiones (Kendall & Kendall, 2005, pág. 3).

- **Sistemas de Apoyo en la Toma de Decisiones (*DSS Decision Support Systems*)** se encuentran entre sistemas de alto nivel. Similares a los MIS, pero con énfasis en la toma de decisiones. Son hechos a la medida de cada tomador de decisiones (Kendall & Kendall, 2005, pág. 3).

- **Sistemas Expertos de Inteligencia Artificial (*ESAI*).** Aplican el conocimiento de los encargados de la toma de decisiones y los enfoques del razonamiento de la inteligencia artificial para solucionar problemas estructurados específicos (Kendall & Kendall, 2005, pág. 3).

  Arquitectura recomendada para este tipo de sistemas

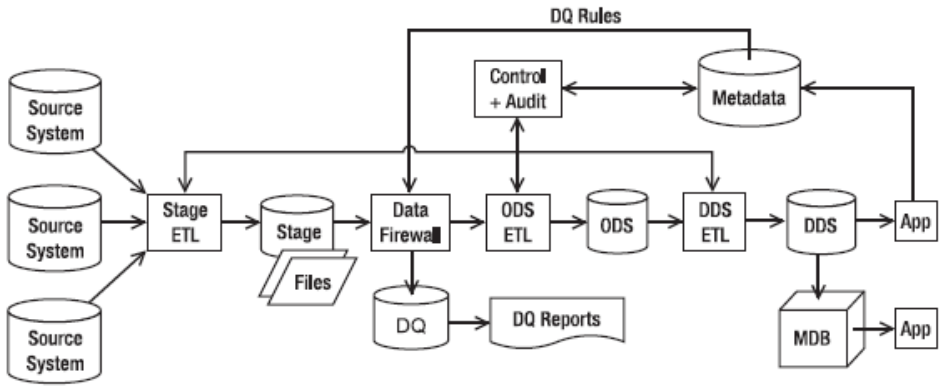


Imagen 11: Arquitectura sistemas BI: Fuente (Rainardi, 2008, pág. 32)

II.1.2.4. SISTEMAS DE GESTIÓN ESTRATÉGICO

Según Kendall k. y Kendall J, (Kendall & Kendall, 2005)Se pueden dividir en dos grupos

- **Sistemas de Apoyo en la toma de Decisiones en grupo (*GDSS Competer-Supported Collaborative Work Systems*).** Auxilian la toma de decisiones semi-estructuradas o no estructuradas a nivel de grupo; permiten a un grupo de personas interactuar en un entorno de decisión colaborativa y controlada con el fin de resolver determinados problemas (Kendall & Kendall, 2005, pág. 4).

- **Sistemas de Apoyo a Ejecutivos (*ESS Executive Support Systems*)** se encuentran en el nivel estratégico de la administración, ayudan a los ejecutivos a organizar sus interacciones con el ambiente externo proporcionando datos resumidos, indicadores, gráficos, etc. Estos se apoyan en la información generada por los TPS y los MIS y ayudan a las decisiones no estructuradas (Kendall & Kendall, 2005, pág. 4).

- **Sistemas de Trabajo Corporativo Apoyados por Computadora (*CSCWS, Computer-Supported Collaborative Work Systems*),** descritos de manera más general, auxilian la toma de decisiones semiestructuradas o no estructuradas a nivel de grupo (Kendall & Kendall, 2005, pág. 4).

  La Arquitectura recomendada es similar para sistemas de gestión táctico

II.1.3. DESCUBRIMIENTO DE CONOCIMIENTO EN BASES DE DATOS (*KDD*)

En los últimos años, ha existido un gran crecimiento en nuestras capacidades de generar y colectar datos, debido básicamente al gran poder de procesamiento de las máquinas como a su bajo costo de almacenamiento.

Sin embargo, dentro de estas enormes masas de datos existe una gran cantidad de información oculta, de gran importancia estratégica, a la que no se puede acceder por las técnicas clásicas de recuperación de la información.

El descubrimiento de esta información oculta es posible gracias a la minería de datos (*Data Mining*), que entre otras sofisticadas técnicas aplica la inteligencia artificial para encontrar patrones y relaciones dentro de los datos permitiendo la creación de modelos, es decir, representaciones abstractas de la realidad, pero es el descubrimiento del conocimiento (*KDD*, por sus siglas en inglés) que se encarga de la preparación de los datos y la interpretación de los resultados obtenidos, los cuales dan un significado a estos patrones encontrados.

Así el valor real de los datos reside en la información que se puede extraer de ellos, información que ayude a tomar decisiones o mejorar nuestra comprensión de los fenómenos que nos rodean. Hoy, más que nunca, los métodos analíticos avanzados son el arma secreta de muchos negocios exitosos.

Empleando métodos analíticos avanzados para la explotación de datos, los negocios incrementan sus ganancias, maximizan la eficiencia operativa, reducen costos y mejoran la satisfacción del cliente.

II.1.3.1. CONCEPTO DEL *KDD*

De forma general, los datos son la materia prima bruta. En el momento que el usuario les atribuye algún significado especial pasan a convertirse en información.

Cuando los especialistas elaboran o encuentran un modelo, haciendo que la interpretación de la información y ese modelo representen un valor agregado, entonces nos referimos al conocimiento. En la imagen 12 se ilustra la jerarquía que existe en una base de datos entre datos, información y conocimiento. Se observa igualmente el volumen que presenta en cada nivel y el valor que los responsables de las decisiones le dan en esa jerarquía.

El área interna dentro del triángulo representa los objetivos que se han propuesto. La separación del triángulo representa la estrecha unión entre dato e información, no así entre la información y el conocimiento.

Imagen 12 Jerarquía del conocimiento: Fuente: Propia

La capacidad de generar y almacenar información creció considerablemente en los últimos tiempos, se ha estimado que la cantidad de datos en el mundo almacenados en bases de datos se duplica cada 20 meses. Es así que hoy las organizaciones tienen gran cantidad de datos almacenados y organizados, pero a los cuales no les pueden analizar eficientemente en su totalidad. Con las sentencias SQL se puede realizar un primer análisis, aproximadamente el 80% de la información se obtiene con estas técnicas. El 20% restante, que la mayoría de las veces, contiene la información más importante, requiere la utilización de técnicas más avanzadas. El descubrimiento de conocimiento en bases de datos (*KDD*) apunta a procesar automáticamente grandes cantidades de datos para encontrar conocimiento útil en ellos, de esta manera permitirá al usuario el uso de esta información valiosa para su conveniencia.

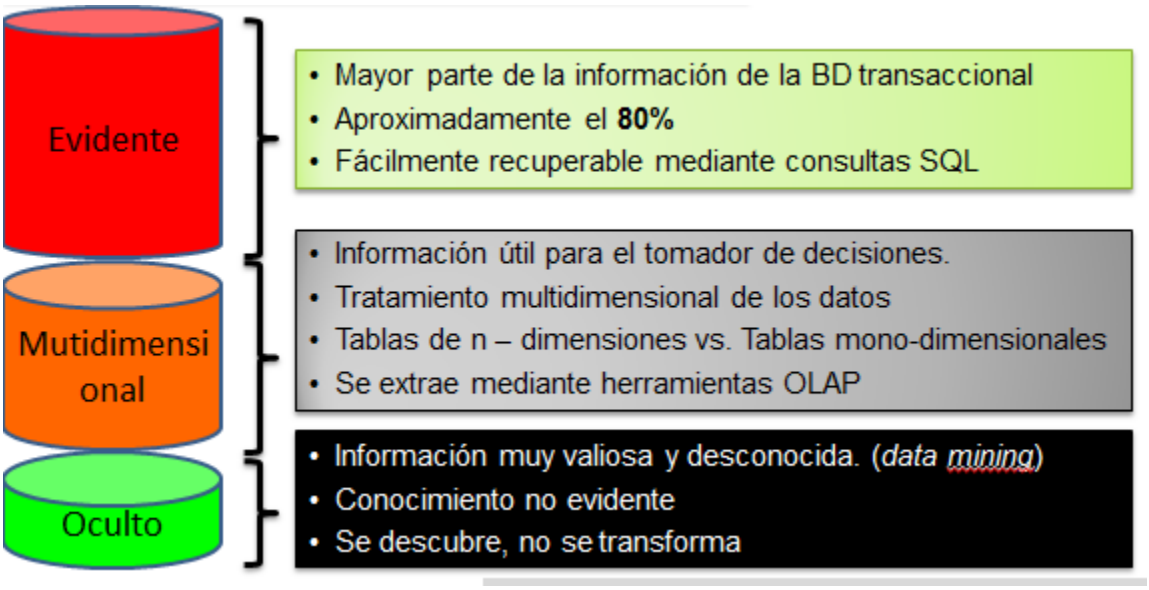


Imagen 13 Análisis y descubrimiento de datos en información: Fuente: Propia

El *KDD* es el proceso no trivial de identificar patrones válidos, novedosos, potencialmente útiles y, en última instancia, comprensibles a partir de los datos. El objetivo fundamental del *KDD* es encontrar conocimiento útil, válido,

Relevante y nuevo sobre un fenómeno o actividad mediante algoritmos eficientes, dadas las crecientes órdenes de magnitud en los datos. Al mismo tiempo hay un profundo interés por presentar los resultados de manera visual o al menos de manera que su interpretación sea muy clara. Otro aspecto es que la interacción humano-máquina deberá ser flexible, dinámica y colaboradora. El resultado de la exploración deberá ser interesante y su calidad no debe ser afectada por mayores volúmenes de datos o por ruido en los datos. En este sentido, los algoritmos de descubrimiento de información deben ser altamente robustos.

- Metas.- Las metas del *KDD* son:
    - Procesar automáticamente grandes cantidades de datos crudos.
    - Identificar los patrones más significativos y relevantes.

- Presentarlos como conocimiento apropiado para satisfacer las metas del usuario.

II.1.3.2. RELACIÓN CON OTRAS DISCIPLINAS

- *KDD* nace como interfaz y se nutre de diferentes disciplinas:
- Sistemas de información/bases de datos: tecnologías de bases de datos y bodegas de datos, maneras eficientes de almacenar, accesar y manipular datos.
- Estadística, aprendizaje automático/IA (redes neuronales, lógica difusa, algoritmos genéticos, razonamiento probabilístico): desarrollo de técnicas para extraer conocimiento a partir de datos.
- Reconocimiento de patrones: desarrollo de herramientas de clasificación.
- Visualización de datos: interfaz entre humanos y datos, y entre humanos y patrones.
- Computación paralela/distribuida: cómputo de alto desempeño, mejora de desempeño de algoritmos debido a su complejidad y a la cantidad de datos.
- Interfaces de lenguaje natural a bases de datos.

II.1.3.3. EL PROCESO DEL *KDD*

El proceso de *KDD* consiste en usar métodos de minería de datos (algoritmos) para extraer (identificar) lo que se considera como conocimiento de acuerdo a la especificación de ciertos parámetros usando una base de datos junto con pre-procesamientos y post-procesamientos. En la imagen 14 se ilustra el proceso de *KDD*.

Se estima que la extracción de patrones (minería) de los datos ocupa solo el 15% al 20% del esfuerzo total del proceso de *KDD*.

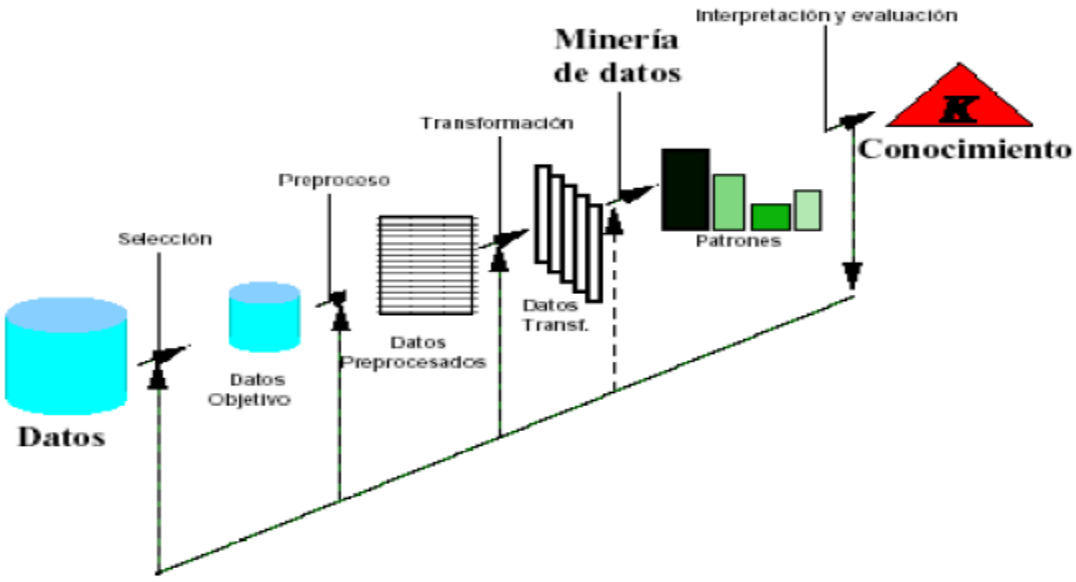


Imagen 14: Proceso del KDD: Fuente: Propia

El proceso de descubrimiento de conocimiento en bases de datos involucra varios pasos:

- Determinar las fuentes de información: que pueden ser útiles y dónde conseguirlas.

- Diseñar el esquema de un almacén de datos (*Data Warehouse*): que consiga unificar de manera operativa toda la información recogida.

- Implantación del almacén de datos: que permita la navegación y visualización previa de sus datos, para discernir qué aspectos puede interesar que sean estudiados. Esta es la etapa que puede llegar a consumir el mayor tiempo.

- Selección, limpieza y transformación de los datos que se van a analizar: la selección incluye tanto una criba o fusión horizontal (filas) como vertical (atributos). La limpieza y pre-procesamiento de datos se logra

diseñando una estrategia adecuada para manejar ruido, valores incompletos, secuencias de tiempo, casos extremos (si es necesario), etc.

- Seleccionar y aplicar el método de minería de datos apropiado: esto incluye la selección de la tarea de descubrimiento a realizar, por ejemplo, clasificación, agrupamiento o *clustering*, regresión, etc. La selección de él o de los algoritmos a utilizar. La transformación de los datos al formato requerido por el algoritmo específico de minería de datos. Y llevar a cabo el proceso de minería de datos, se buscan patrones que puedan expresarse como un modelo o simplemente que expresen dependencias de los datos, el modelo encontrado depende de su función (clasificación) y de su forma de representarlo (árboles de decisión, reglas, etc.), se tiene que especificar un criterio de preferencia para seleccionar un modelo dentro de un conjunto posible de modelos, se tiene que especificar la estrategia de búsqueda a utilizar (normalmente está predeterminada en el algoritmo de minería)

- Evaluación, interpretación, transformación y representación de los patrones extraídos:

  Interpretar los resultados y posiblemente regresar a los pasos anteriores. Esto puede involucrar repetir el proceso, quizás con otros datos, otros algoritmos, otras metas y otras estrategias. Este es un paso crucial en donde se requiere tener conocimiento del dominio. La interpretación puede beneficiarse de procesos de visualización, y sirve también para borrar patrones redundantes o irrelevantes.

- Difusión y uso del nuevo conocimiento.

  Incorporar el conocimiento descubierto al sistema (normalmente para mejorarlo) lo cual puede incluir resolver conflictos potenciales con el conocimiento existente.

El conocimiento se obtiene para realizar acciones, ya sea incorporándolo dentro de un sistema de desempeño o simplemente para almacenarlo y reportarlo a las personas interesadas.

En este sentido, *KDD* implica un proceso interactivo e iterativo involucrando la aplicación de varios algoritmos de minería de datos.

II.1.4. BABOK

La guía sobre los fundamentos del conocimiento del análisis de negocio es un estándar reconocido mundialmente para la práctica del análisis de negocio. La guía *BABOK*® describe las áreas de conocimiento del análisis de negocio, sus actividades, tareas asociadas y las habilidades necesarias para ser eficaz en su realización.

El propósito primordial de la guía *BABOK*® es definir la profesión del análisis de negocio. Sirve como un punto de referencia acordado por los profesionales para poder discutir sobre el trabajo que realizan y asegurar que tienen las habilidades necesarias para cumplir con el rol de manera eficiente; y define las habilidades y conocimiento que las personas que trabajan con analistas de negocio y emplean analistas de negocio deben esperar de un profesional experimentado. Es el marco de referencia que describe las tareas del análisis de negocio que se deben realizar con el fin de entender como una solución ofrecerá valor a una organización patrocinadora. Las formas que toman esas tareas, el orden en que son llevadas a cabo, la importancia relativa de las tareas y otros aspectos pueden variar, pero cada tarea contribuye de alguna forma, directa o indirectamente a la meta general. (IIBA_BABOKv2, 2009, pág. 9)

En la presente tesis sólo se emplearan las siguientes tareas:

- Documentar los resultados de la ‘elicitación’
- Confirmar los resultados de la ‘elicitación’

- Determinar el enfoque de la solución
- Definir el caso de negocio
- Priorizar los requerimientos
- Especificar y modelar los requerimientos
- Evaluar la solución propuesta

II.1.5. PMBOK

La guía de los fundamentos para la dirección de proyectos (guía del *PMBOK®)* Quinta Edición) proporciona pautas para la dirección de proyectos individuales y define conceptos relacionados con la dirección de proyectos. Describe asimismo el ciclo de vida de la dirección de proyectos y los procesos relacionados, así como el ciclo de vida del proyecto.

La guía del *PMBOK*® contiene el estándar, reconocido a nivel global y la guía para la profesión de la dirección de proyectos. Por estándar se entiende un documento formal que describe normas, métodos, procesos y prácticas establecidos. Al igual que en otras profesiones, el conocimiento contenido en este estándar evolucionó a partir de las buenas prácticas reconocidas de los profesionales dedicados a la dirección de proyectos que han contribuido a su desarrollo. (PMI_PMBOKv5, 2013, pág. 28)

En la presente tesis sólo se emplearan los siguientes procesos.

- Monitorear y controlar el trabajo del proyecto
- Cerrar proyecto o fase
- Definir el alcance
- Controlar el cronograma

- Estimar los costos
- Determinar el presupuesto
- Controlar los costos

II.1.6. LÉXICO EXTENDIDO DEL LENGUAJE (*LEL*)

II.1.6.1. LÉXICO EXTENDIDO DEL LENGUAJE Y ESCENARIOS

Una cuestión fundamental en el desarrollo de software es establecer una buena comunicación entre los distintos participantes. Uno de los obstáculos para la comunicación es el uso de diferentes léxicos por parte del ingeniero de requerimientos y el usuario. Por lo tanto, el uso de un vocabulario común, en particular el uso del lenguaje propio del usuario, mejora considerablemente esta comunicación (Bertolami, 2010, págs. 7-11) (Hadad, 2008)

El léxico extendido del lenguaje (*LEL*) y los escenarios son herramientas para la Elicitación de Requerimientos y pueden ser usadas a lo largo del ciclo completo de desarrollo de software (Bertolami, 2010, págs. 7-11) . Este enfoque presenta una metodología basada en el uso de LEL para registrar el vocabulario del macrosistema y escenarios para modelar el comportamiento. Ambas herramientas presentan la ventaja de utilizar el lenguaje natural para las descripciones, lo que favorece la comunicación y validación con el usuario (Bertolami, 2010, págs. 7-11) (Hadad, 2008).

- **LÉXICO EXTENDIDO DEL LENGUAJE (*LEL*)**

El léxico extendido del lenguaje es un meta modelo diseñado para facilitar la elicitación y representación del lenguaje usado en la aplicación (Bertolami, 2010, págs. 7-11). El *LEL* tiene como finalidad la comprensión del vocabulario de la aplicación, sin considerar en esta etapa la comprensión del problema. Se propone construir el *LEL* como primer paso en la fase de elicitación de requerimientos, tratando de conocer el vocabulario que utiliza el usuario en su mundo real. El principal

propósito de *LEL* es capturar el vocabulario y la semántica de la aplicación, posponiendo la comprensión de la funcionalidad de la aplicación (Bertolami, 2010, págs. 7-11). Una vez que se está familiarizado con ese léxico, la comunicación con el usuario y la comprensión del problema tendrá un obstáculo menos: el vocabulario (Bertolami, 2010, págs. 7-11) (Hadad, 2008).

El *LEL* es una representación hipertextual de los símbolos del lenguaje del cliente en el contexto de la aplicación. Cada símbolo representa una palabra o frase, frecuentemente las más repetidas por el cliente o de importancia relevante para el sistema, excluyendo aquellas palabras o frases que son de conocimiento general, las que son demasiado específicas y las que tienen un significado muy amplio (Bertolami, 2010, págs. 7-11) (Hadad, 2008).

Para conocer el vocabulario propio del dominio del sistema, el ingeniero de software planifica alguna estrategia para recolectar información, principalmente entrevistas con el usuario, las que eventualmente pueden ser reemplazadas por documentos descriptivos de la aplicación. En esta etapa se registran las frases o palabras usadas con más frecuencia o que tienen un significado especial. Como resultado de fase se obtiene una lista candidata formada por símbolos o entradas candidatos, que será utilizada como base para realizar nuevas entrevistas con el cliente.

Los símbolos se clasifican en las categorías generales sujeto (entidad activa que realiza actividades en el dominio de la aplicación), Objeto (entidad pasiva a la que se le aplican acciones en el dominio de la aplicación, sin realizar acciones por sí misma),

Verbo (actividad o acción que ocurre en el dominio de la aplicación) y estado (condición o situación en la cual sujetos, objetos o verbos del dominio de la aplicación están o pueden estar en un momento dado). En caso de ser necesario puede considerarse refinar algunos tipos de la

clasificación general o crear algunos nuevos, ajustando la clasificación al dominio específico de la aplicación (Bertolami, 2010, págs. 7-11). (Hadad, 2008)

A partir del conocimiento obtenido, cada símbolo se describe en términos de nombre, noción e impacto. El nombre identifica el símbolo, la noción de nota lo que significa el símbolo y el impacto (puede no existir) cómo repercute en el sistema. Al describir noción e impacto deben tenerse en cuenta dos principios: circularidad que implica maximizar el uso de los símbolos en el significado de otros símbolos y vocabulario mínimo que significa minimizar el uso de símbolos externos al lenguaje de la aplicación. La imposición de ambos principios da como resultado la formación de una red de elementos vinculados entre sí, que se representa como un documento de hipertexto. Para conocer todo el vocabulario del dominio se puede navegar a través del hipertexto (Bertolami, 2010, págs. 7-11) (Hadad, 2008)

La lista candidata es validada con el cliente con el fin de rectificar o ratificar las descripciones. Durante esta etapa puede haber modificaciones en los símbolos, incorporación de nuevos símbolos y eliminación de otros. Este proceso iterativo de interacción con el cliente permite refinar la lista hasta obtener la lista definitiva. Forma parte de esta etapa de validación el desarrollo de escenarios.

LEL: representación de los símbolos en el lenguaje del dominio de la aplicación.
Sintaxis:
$\{Símbolo\}_1^N$

Símbolo: entrada del léxico que tiene un significado especial en el dominio de la aplicación.
Sintaxis:
$\{Nombre\}_1^N + \{Noción\}_1^N + \{Impacto\}_1^N$

Nombre: identificación del símbolo. Más de uno representa sinónimo.
Sintaxis:
Palabra | Frase

Noción: denotación del símbolo. Debe ser expresada usando referencias a otros símbolos y usando un vocabulario mínimo.
Sintaxis:
Sentencia

Impacto: connotación del símbolo. Debe ser expresada usando referencias a otros símbolos y usando un vocabulario mínimo.
Sintaxis:
Sentencia

donde Sentencia está formada por Símbolos y No-Símbolos, los últimos pertenecen al vocabulario mínimo.

+ significa composición, {x} significa cero o más ocurrencias de x, | significa **or**

Imagen 15: Modelo del LEL Fuente: (Bertolami, 2010)

El siguiente es un ejemplo de un símbolo del *LEL* (el texto subrayado representa un símbolo del *LEL*).

no show
- Notion
  - Es la baja de la *solicitud de reserva* por no presentación del *pasajero*.
  - Es consignado por el *recepcionista*.
  - Se hace en la *recepción*.
- Behavioral responses
  - **If** el *pasajero* no se presenta entre las 12 hrs. la fecha indicada en la *solicitud de reserva* y las 06 hrs. del día siguiente **then** se elimina la *solicitud de reserva* en la *planilla de reservas*.
  - Se actualiza la *disponibilidad de habitaciones* en la *planilla de ocupación de habitaciones*.

Imagen 16: Símbolo del LEL Fuente: (Bertolami, 2010)

- **ESCENARIOS**

Los enfoques basados en escenarios son ampliamente usados en la comunidad de los Sistemas de Información (Bertolami, 2010, págs. 7-11) (Potts & Antón, 1998) (Sutcliffe, 1997) (Weidenhaupt, Pohl, Jarke, & Haumer, 1998). Los escenarios son descripciones parciales del comportamiento del sistema y su entorno en situaciones particulares.

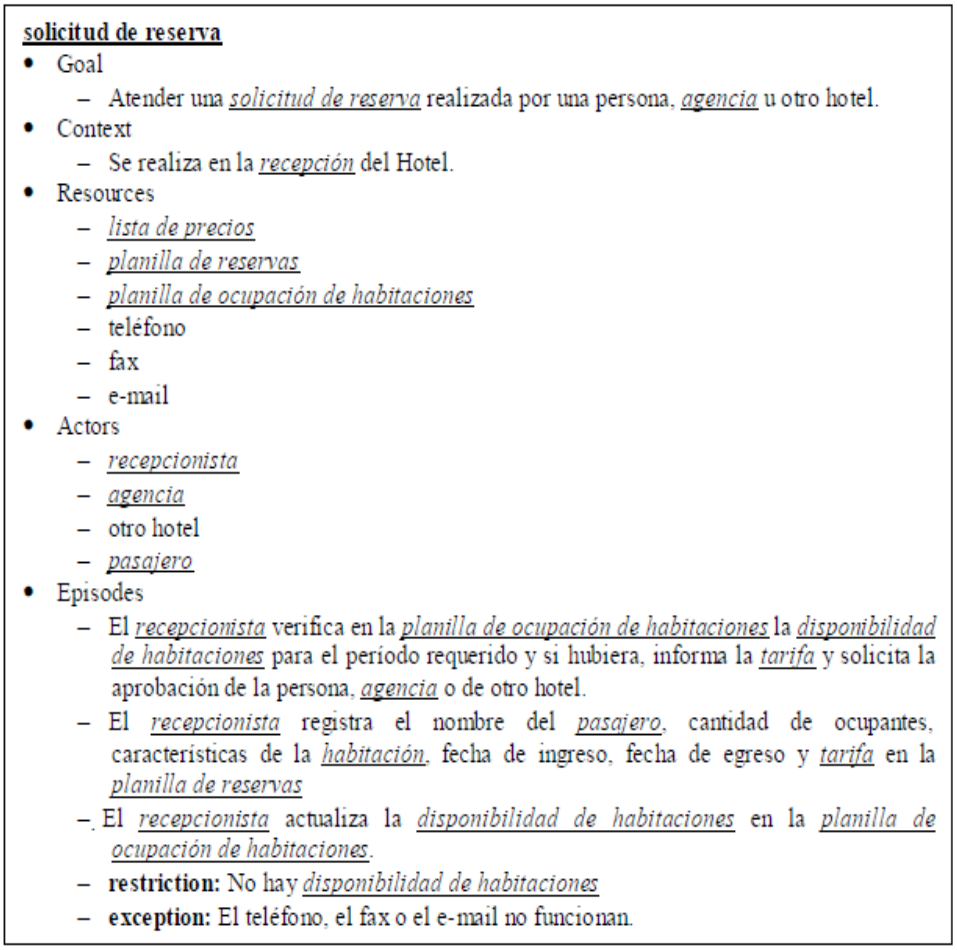
Pueden usarse para representar el comportamiento de sistemas existentes, aunque más comúnmente son usados durante la planificación y diseño de nuevos sistemas. Los escenarios desempeñan un importante rol a través del proceso de desarrollo de Sistemas de Información, pues son fundamentales para las siguientes actividades: describir y aclarar las propiedades relevantes del dominio de la aplicación, descubrir los requerimientos del sistema, evaluar alternativas de diseño, validar diseños (Bertolami, 2010, págs. 7-11) (Benner, Feather, Johnson, & Zorman, 1993), facilitar la comunicación entre los participantes, favorecer el acuerdo entre los requerimientos del usuario y el soporte provisto por el sistema (Bertolami, 2010, págs. 7-11) (Sutcliffe, 1997), facilitar la comprensión de las prácticas de trabajo y los procesos del negocio, representar requerimientos de comportamiento, validar requerimientos, generar y validar diseños orientados a objetos, generar y revisar casos de prueba de sistemas (Bertolami, 2010, págs. 7-11) (Potts & Antón, 1998).

Los escenarios son usados para entender la aplicación y su funcionalidad cada escenario describe una situación específica de la aplicación centrando la atención en su comportamiento. Los escenarios se representan en lenguaje natural, son derivados desde el *LEL* mediante heurísticas específicas, evolucionan durante el proceso de construcción del software y se representan de manera estructurada (Bertolami, 2010, págs. 7-11).

LEL y escenarios (*L&E*) constituyen el producto resultante de la fase de Elicitación de Requerimientos. Los escenarios no son especificaciones ni requerimientos, son descripciones auxiliares para el proceso de definición de requerimientos. Ellos proveen una fuente de conocimiento donde pueden ser encontrados los requerimientos y en los que pueden basarse las especificaciones (Bertolami, 2010, págs. 7-11).

Un escenario se compone de nombre, objetivo, contexto, recursos, actores y episodios. Ver plantilla ANEXO C: MODELO DE ESCENARIOS *LEL*

A continuación se reproduce un ejemplo de escenario.



solicitud de reserva

- Goal
  - Atender una *solicitud de reserva* realizada por una persona, *agencia* u otro hotel.
- Context
  - Se realiza en la *recepción* del Hotel.
- Resources
  - *lista de precios*
  - *planilla de reservas*
  - *planilla de ocupación de habitaciones*
  - teléfono
  - fax
  - e-mail
- Actors
  - *recepcionista*
  - *agencia*
  - otro hotel
  - *pasajero*
- Episodes
  - El *recepcionista* verifica en la *planilla de ocupación de habitaciones* la *disponibilidad de habitaciones* para el período requerido y si hubiera, informa la *tarifa* y solicita la aprobación de la persona, *agencia* o de otro hotel.
  - El *recepcionista* registra el nombre del *pasajero*, cantidad de ocupantes, características de la *habitación*, fecha de ingreso, fecha de egreso y *tarifa* en la *planilla de reservas*
  - El *recepcionista* actualiza la *disponibilidad de habitaciones* en la *planilla de ocupación de habitaciones*.
  - **restriction:** No hay *disponibilidad de habitaciones*
  - **exception:** El teléfono, el fax o el e-mail no funcionan.

Imagen 17: Ejemplo de escenarios. Fuente: (Bertolami, 2010)

III. HIPÓTESIS Y OPERACIONALIZACIÓN DE LAS VARIABLES.

III.1. HIPÓTESIS GENERAL

El diseño y puesta en marcha de una metodología integral que permita la estimación funcional de proyectos de *software*, contribuye a mejorar la gestión de proyectos de *software*.

III.2. HIPÓTESIS ESPECÍFICAS

H1.- La utilización de la estimación funcional sobre una metodología integral de estimación, basada en el descubrimiento de conocimiento de datos contribuye en la eficiencia de la gestión de proyectos de *software* en las organizaciones bajo estudio.

H2.- La utilización de la estimación funcional sobre una metodología integral de estimación, basada en el descubrimiento de conocimiento de datos contribuye en la eficacia de la gestión de proyectos de *software* en las organizaciones bajo estudio.

H3.- La utilización de la estimación funcional sobre una metodología integral de estimación, basada en el descubrimiento de conocimiento de datos contribuye en la efectividad de la gestión de proyectos de *software* en las organizaciones bajo estudio.

III.3. VARIABLES

III.3.1. VARIABLES INDEPENDIENTE

- Estimación funcional

III.3.2. VARIABLES INTERVINIENTE

- Metodología de estimación de proyectos basada en el descubrimiento de conocimiento de datos

III.3.3. VARIABLE DEPENDIENTE

- Gestión de proyectos de *software*.

III.3.4. OPERACIONALIDAD DE LAS VARIABLES.

VARIABLES		DIMENSIONES	INDICADORES	TÉCNICAS
INDEPENDIIENTE	Estimación funcional	Puntos función creación	Cantidad ILF Complejidad Alta Creación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad ILF Complejidad Media Creación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad ILF Complejidad Baja Creación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad EIF Complejidad Alta Creación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad EIF Complejidad Media Creación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad EIF Complejidad Baja Creación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad EI Complejidad Alta Creación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad EI Complejidad Media Creación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad EI Complejidad Baja Creación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad EO Complejidad Alta Creación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad EO Complejidad Media Creación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad EO Complejidad Baja Creación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad EQ Complejidad Alta Creación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad EQ Complejidad Media Creación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad EQ Complejidad Baja Creación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)

		Puntos función modificación	Cantidad ILF Complejidad Alta Modificación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad ILF Complejidad Media Modificación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad ILF Complejidad Baja Modificación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad EIF Complejidad Alta Modificación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad EIF Complejidad Media Modificación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad EIF Complejidad Baja Modificación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad EI Complejidad Alta Modificación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad EI Complejidad Media Modificación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad EI Complejidad Baja Modificación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad EO Complejidad Alta Modificación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad EO Complejidad Media Modificación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad EO Complejidad Baja Modificación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad EQ Complejidad Alta Modificación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad EQ Complejidad Media Modificación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
			Cantidad EQ Complejidad Baja Modificación	Técnicas de IFPUG 4.3.1 y SMC RICEF. (Observación Estructurada4)
DEPENDIENTES	Gestión de proyectos	Tiempo del proyecto	Tiempo real del proyecto.	Observación Estructurada

		Costo del proyecto	Monto real del proyecto.	Observación Estructurada
		Desempeño según el tiempo	Variación del tiempo estimado con el real consumido (VTERC). Ver definición.	Análisis de Variación
			Índice del tiempo estimado con el real consumido del proyecto (ITERC). Ver definición.	Análisis de Variación
		Desempeño Según el cronograma	Variación del Cronograma (SV). Ver definición.	Gestión del Valor Ganado
			Desempeño según cronograma (SPI). Ver definición.	Gestión del Valor Ganado
		Desempeño Según el costo	Variación del Costo (CV). Ver definición.	Gestión del Valor Ganado
			Desempeño según Costo (CPI). Ver definición.	Gestión del Valor Ganado
INTERVINIENTE	Metodología de estimación de proyectos basada en el descubrimiento de conocimiento de datos	Entrega oportuna de los proyectos	Cantidad de Proyectos por objetivo y fase	Observación Estructurada
			Grado de cumplimiento del proyecto por objetivo y fase de estimación.	Análisis de Variación
		Productividad del Equipo del Proyecto	Cantidad de PF por tipo de proyecto y tecnología empleada.	Observación Estructurada
			Cantidad de Horas por tipo de proyecto y tecnología empleada.	Observación Estructurada

- **DEFINICIÓN DE INDICADORES EVALUADOS CON LA HERRAMIENTA DE ANÁLISIS DE VARIACIÓN**

1. **Variación del tiempo del proyecto *VTERC***

   La fórmula para calcularlo es ***TE*** menos el ***RC***

2. **Índice según el tiempo del proyecto *ITERC***

   La fórmula es ***TE*** entre el ***RC*** consumido, porcentaje (0% - 100%)

   Dónde:

   TE = Tiempo estimado.- Representa el valor obtenido mediante la plantilla de estimación, empleando las técnicas de *IFPUG 4.3.1* y *SMC RICEF*.:

   RC = Real consumido.- Es el tiempo real consumido en el proyecto al cerrar el proyecto o fase.

   Análisis

Si	TE > RC	TE = RC	TE < RC
Entonces	VTERC < 0	VTERC = 0	VTERC > 0
	ITERC < 1	ITERC = 1	ITERC > 1
El Tiempo estuvo:	Mayor al real consumido	Dentro del tiempo real consumido	Menor al real consumido

3. **Variación del cronograma *SV***

   La fórmula para calcularlo es ***EV*** menos ***PV***

4. **Desempeño según cronograma (*SPI*)**

   La fórmula para calcularlo es ***EV*** dividido entre ***PV***, porcentaje (0% - 100%)

   Dónde:

PV = Valor planificado: Representa el costo planificado del trabajo que debería estar completo en un momento determinado.

EV = Valor ganado: Es una medida del valor del trabajo que se completó a un momento determinado.

Análisis

Si	PV > EV	PV = EV	PV < EV
Entonces	SV < 0	SV = 0	SV > 0
	SPI < 1	SPI = 1	SPI > 1
El Proyecto estuvo	Atrasado	Dentro del cronograma	Adelantado al cronograma

5. Variación del costo *CV*

La fórmula para calcularlo es ***EV*** menos ***AC***

6. Desempeño según costo(*CPI*)

Índice del desempeño del costo. Su fórmula de cálculo es el ***EV*** dividido el entre ***AC***, porcentaje (0% - 100%)

Dónde:

EV = Valor ganado.- Es una medida del valor del trabajo que se completó a un momento determinado.

AC = Costo actual.- Representa el monto que gastamos para completar el trabajo.

Análisis:

Si	AC > EV	AC = EV	AC < EV
Entonces	CV < 0	CV = 0	CV > 0
	CPI < 1	CPI = 1	CPI > 1
El presupuesto estuvo	Sobre el presupuesto	Dentro del presupuesto	Debajo del presupuesto

III.4. TIPO

Por la naturaleza de la situación problemática que se pretende resolver, el propósito y el objetivo planteado, la presente tesis debe ser considerada como un trabajo de investigación aplicada y correlacional. Para la gestión de información se construirá una metodología integral que permita la estimación de proyectos de *software*, desde la elicitación de requisitos (*BABOK*), hasta la planificación del proyecto (*PMBOK*) complementada con las técnicas de estimación funcional (*UFP, SMC/RICEF*) y el léxico extendido del lenguaje (*LEL*), que sea retroalimentada por el descubrimiento de conocimiento de datos (*KDD*) de las casuísticas estructuradas presentadas en los proyectos *post morten*, y un aplicativo que monitorea el comportamiento de los proyectos en las etapas de viabilidad, análisis, diseño, planificación y cierre de los mismos.

III.5. POBLACIÓN:

Todas las empresas u organizaciones que desean realizar desarrollo de aplicaciones de *Software*. Especialmente las empresas dedicadas al desarrollo de *Software*.

III.6. UNIVERSO SOCIAL:

Profesionales: Ingenieros de Sistemas y carrera vinculadas al desarrollo de soluciones de *software*

Investigadores: en temas de desarrollo de soluciones de *software*

III.7. MUESTRA:

Todos los proyectos y evolutivos de pequeño tamaño realizados por EDELNOR desde el 2010 hasta el 2016 y STEFANINI desde el 2014 hasta el 2016.

IV. MÉTODO.

IV.1. DISEÑO DE INVESTIGACIÓN

Para el desarrollo de la tesis se realizarán las siguientes actividades:

1. **Automatizar la metodología de estimación del *IFPUG 4.3.1* y *SMC RICEF*.-** En esta fase se realizará la automatización de las técnicas de estimación de *IFPUG 4.3.1* y *SMC RICEF*, para la toma de datos. Empleando para ello las herramientas *de ASPNET - C# .NET 2013 y SQL 2014*,

2. **Realizar la estimación de los proyectos en la etapa de Viabilidad.-.-** En esta fase se realizará la estimación de proyectos empleando el método del *IFPUG 4.3.1* y *SMC RICEF*

3. **Realizar la estimación de los proyectos en la etapa de Análisis.-.-** En esta fase se realizará la estimación de proyectos empleando el método del *IFPUG 4.3.1* y *SMC RICEF*

4. **Realizar la estimación de los proyectos en la etapa de Diseño.-** En esta fase se realizará la estimación de proyectos empleando el método del *IFPUG 4.3.1* y *SMC RICEF*

5. **Realizar la estimación de los proyectos operativos en la etapa de planificación del proyecto.-** En esta fase se realizará la estimación del proyecto empleando el método del *IFPUG 4.3.1* y *SMC RICEF*

6. **Consolidar información de los proyectos.-** En esta fase se realizará la consolidación de la información de los proyectos de gestión operacional, táctico y estratégico.

7. **Diseñar y desarrollar los algoritmos de minería de datos.-** En esta fase se realizará el análisis, diseño y el desarrollo de los algoritmos de minería de datos (*Cluster* y *Neural Network*) empleando para ello la herramienta *SSAS* (*SQL 2014*

Server Analysis Services), para evaluar la casuística de los proyectos dentro y fuera del rango. Por etapas (viabilidad, análisis, diseño y planificación).

8. **Desarrollar una herramienta de monitoreo y control de proyectos.-** En esta fase se desarrollará una herramienta con los indicadores y/o reportes que permitan analizar y tomar las decisiones correspondientes para la gestión proactiva de los proyectos.), empleando para ello las herramientas *SSIS, SSAS y SSRS* (*SQL 2014*),

9. **Evaluar la información de los proyectos.-** En esta fase se realizará la evaluación de la información de los proyectos de *software*.

IV.2. ESTRATEGIA DE PRUEBA DE HIPÓTESIS

Se desarrollará una metodología de procesos y/o tareas de medición en 5 etapas

1. **Viabilidad**

    - Dimensionar requerimientos viabilidad
    - Determinar la estimación de los requerimientos viabilidad

2. **Análisis**

    - Dimensionar requerimientos análisis
    - Determinar la estimación de los requerimientos análisis

3. **Diseño**

    - Dimensionar requerimientos solución propuesta
    - Determinar la estimación de los requerimientos viabilidad

4. **Planificación**

    - Dimensionar los requisitos del alcance del proyecto

- Determinar la estimación de los requisitos del alcance del proyecto

5. Cierre

- Datos de desempeño del trabajo (incluye los puntos función)

Para ello se construirá un aplicativo para capturar los puntos función a crear y/o modificar y la cantidad de horas de los proyectos., y los reportes que permitan analizar el comportamiento de los proyectos en cada etapa. Además se construirá un modelo de minería de datos basado en *Cluster* y *Neural Network* para analizar el comportamiento de los proyectos dentro y fuera del rango.

IV.3. TÉCNICAS DE RECOLECCIÓN DE DATOS

IV.3.1. INSTRUMENTOS DE RECOLECCIÓN DE DATOS.

Los datos serán obtenidos mediante la técnica de observación estructurada en las plantillas de puntos de función empleadas en EDELNOR y STEFANINI.

Los datos a recolectados serán:

i. **Cantidad de puntos de función por proyecto**

ii. **Esfuerzo en tiempo del proyecto.**

iii. **Monto presupuestado del proyecto.**

iv. **Cantidad de *Internal Logical File* (*ILF*) por proyecto**

1. Cantidad *ILF* complejidad alta.
2. Cantidad *ILF* complejidad media.
3. Cantidad *ILF* complejidad baja.

v. Cantidad de *External Interface File* (*EIF*)por proyecto

1. Cantidad *EIF* complejidad alta.
2. Cantidad *EIF* complejidad media.
3. Cantidad *EIF* complejidad baja.

vi. Cantidad de *External Input* (*EI*)) por proyecto

1. Cantidad *EI* complejidad alta.
2. Cantidad *EI* complejidad media.
3. Cantidad *EI* complejidad baja.

vii. Cantidad de *External Output* (*EO*) por proyecto

1. Cantidad *EO* complejidad alta.
2. Cantidad *EO* complejidad media.
3. Cantidad *EO* complejidad baja.

viii. Cantidad de *External Query* (*EQ*) por proyecto

1. Cantidad *EQ* complejidad alta.
2. Cantidad *EQ* complejidad media.
3. Cantidad *EQ* complejidad baja.

Ver plantilla de toma de datos de puntos función ANEXO D:

Las técnicas complementarias para el análisis de datos son:

- Análisis de Variación
- Gestión del Valor Ganado (EVM)
- Minería de Datos (Clúster y Redes Neuronales)

- Observación estructurada

- **METODOLOGÍA PROPUESTA**

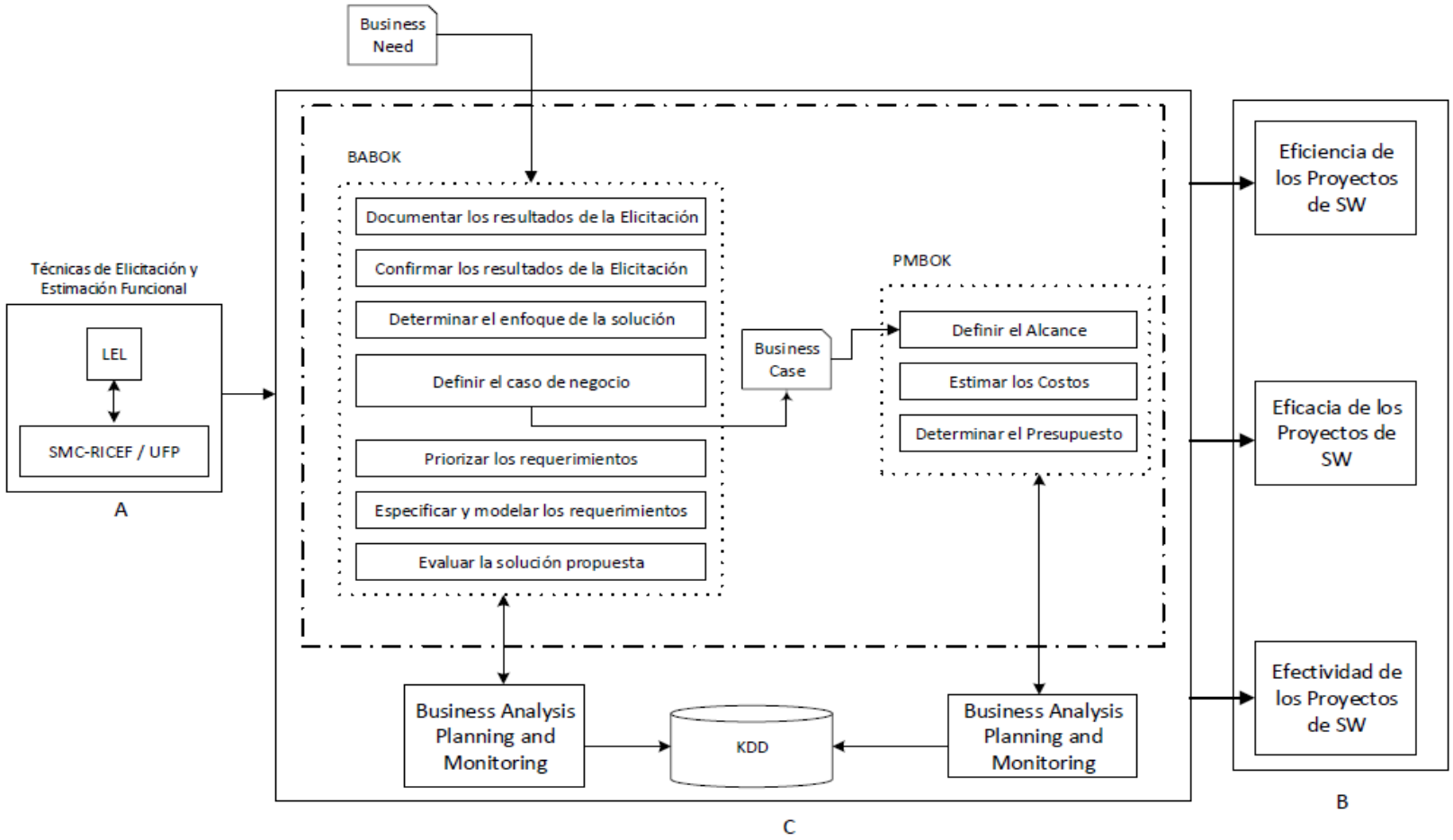
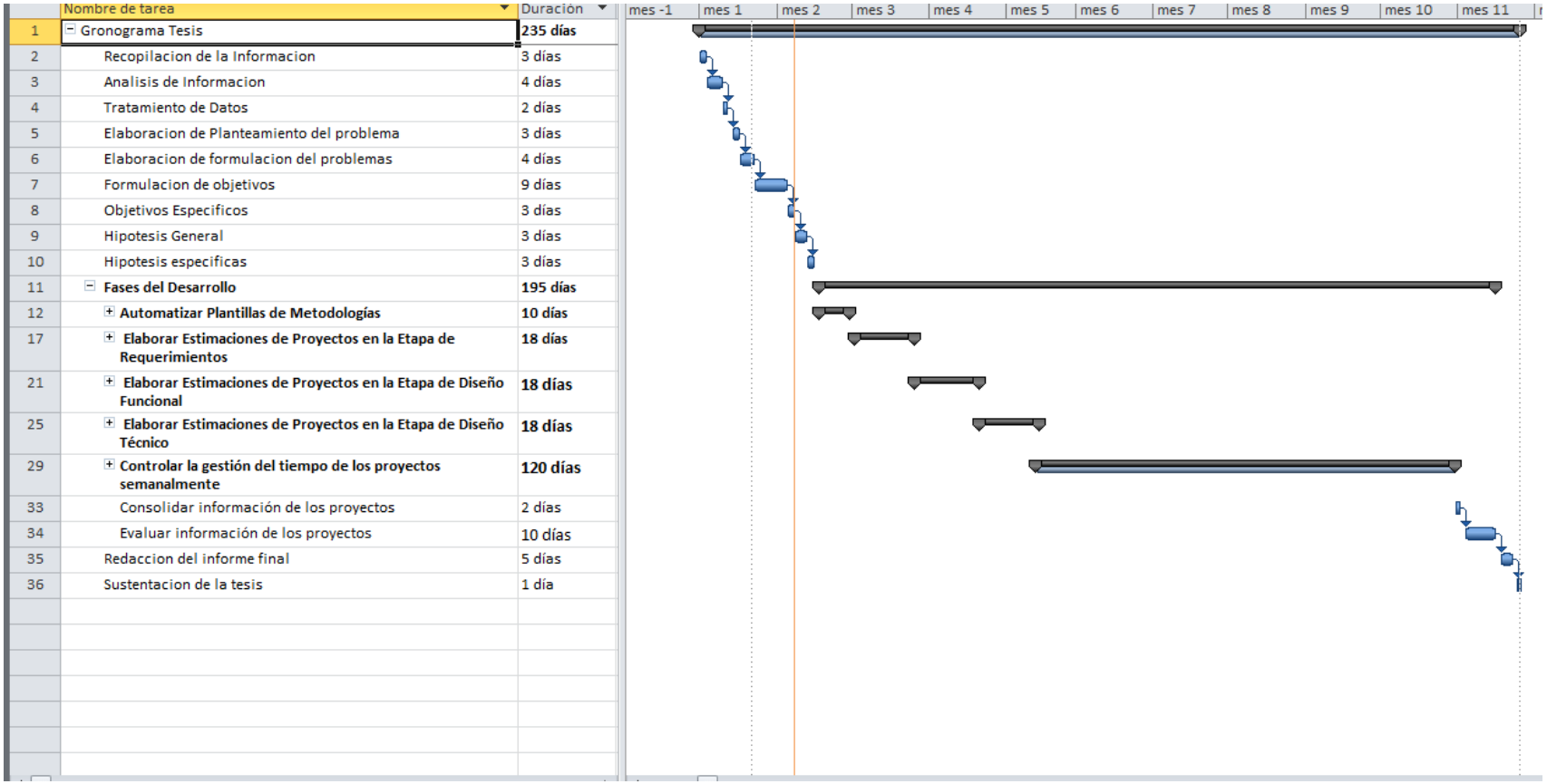


Imagen 18 Propuesta del modelo de estimación de la tesis. Fuente propia.

V. CRONOGRAMA



	Nombre de tarea	Duración
1	Gronograma Tesis	235 días
2	Recopilacion de la Informacion	3 días
3	Analisis de Informacion	4 días
4	Tratamiento de Datos	2 días
5	Elaboracion de Planteamiento del problema	3 días
6	Elaboracion de formulacion del problemas	4 días
7	Formulacion de objetivos	9 días
8	Objetivos Especificos	3 días
9	Hipotesis General	3 días
10	Hipotesis especificas	3 días
11	**Fases del Desarrollo**	**195 días**
12	**Automatizar Plantillas de Metodologías**	**10 días**
17	**Elaborar Estimaciones de Proyectos en la Etapa de Requerimientos**	**18 días**
21	**Elaborar Estimaciones de Proyectos en la Etapa de Diseño Funcional**	**18 días**
25	**Elaborar Estimaciones de Proyectos en la Etapa de Diseño Técnico**	**18 días**
29	**Controlar la gestión del tiempo de los proyectos semanalmente**	**120 días**
33	Consolidar información de los proyectos	2 días
34	Evaluar información de los proyectos	10 días
35	Redaccion del informe final	5 días
36	Sustentacion de la tesis	1 día

Imagen 19.1 Cronograma de tesis. Fuente propia.

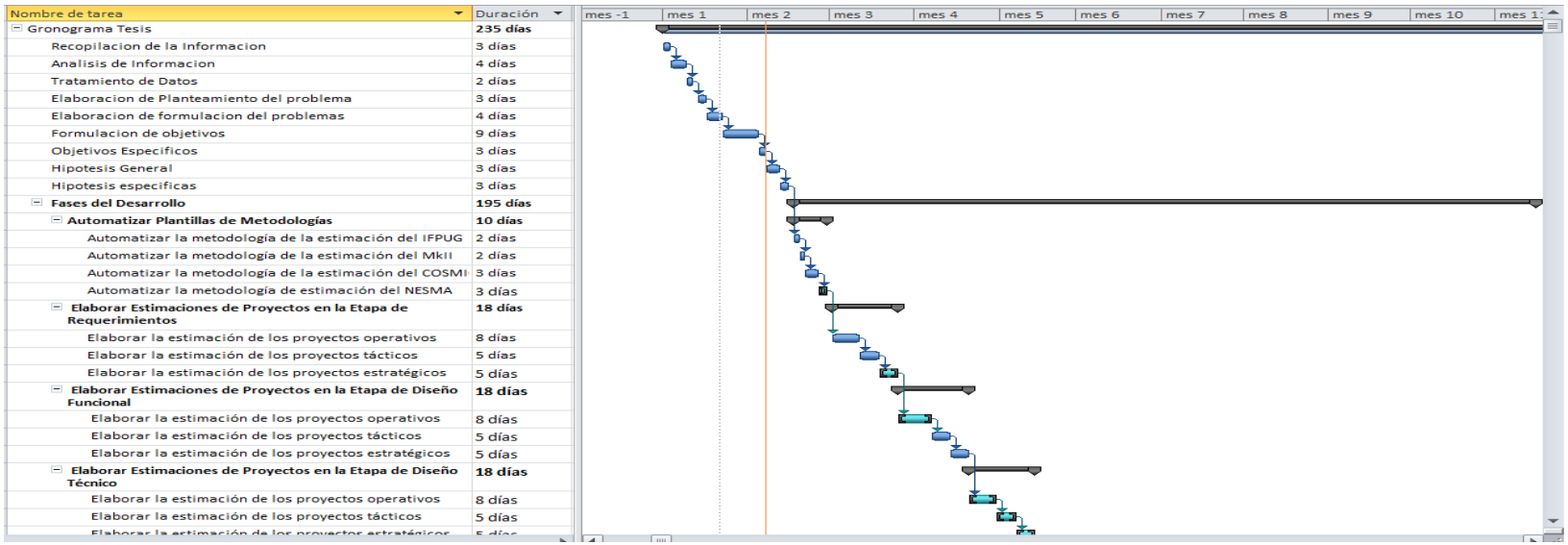


Imagen 19.2 Cronograma de tesis. Fuente propia.

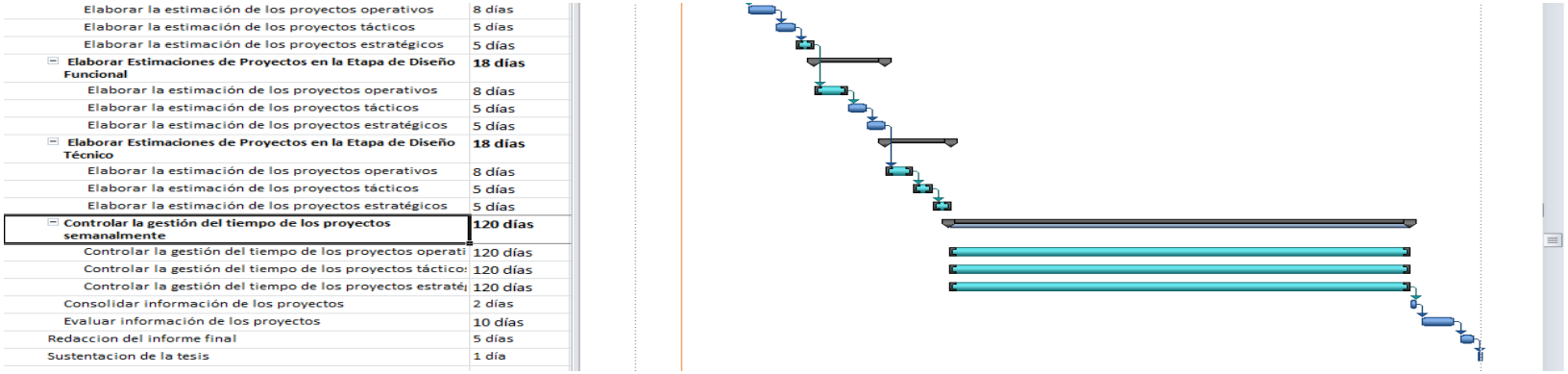


Imagen 19.3 Cronograma de tesis. Fuente propia.

VI. PRESUPUESTO

COSTO DEL PROYECTO		
CONCEPTO		***MONTO S/.***
1. Personal	Encuestador	2,000.00
	Personal de apoyo	800.00
2. Materiales	Útiles de escritorio	1,000.00
3. Equipos de Computo	Pcs	1,500.00
4. Servicios	Luz, Agua, Internet, Copias	800.00
5. Otros Costos	Refrigerio, Transporte, Otros	300.00
TOTAL LINEA BASE		***6,400.00***
6. Reserva de contingencia		500.00
7. Reserva de Gestión		500.00
TOTAL PRESUPUESTO		***7,400.00***

División de los Costos

COSTOS VARIABLES		
CONCEPTO		***MONTO S/.***
1. Personal	Analista	2,000.00
	Personal de Apoyo	800.00
TOTAL COSTO VARIABLE		***2,800.00***

COSTOS FIJOS		
CONCEPTO		***MONTO S/.***
2. Materiales	Útiles de escritorio	1,000.00
3. Equipos de Cómputo	Pcs	1,500.00
4. Servicios	Luz, Agua, Internet, Copias	800.00
5. Otros Costos	Refrigerio, Transporte, Otros	300.00
6. Reserva de contingencia		500.00
7. Reserva de Gestión		500.00
TOTAL COSTO FIJO		***4,600.00***

VII. REFERENCIAS BIBLIOGRÁFICAS

Abran, A. (2009). *The COSMIC Functional Size Measurement Method Version 3.0.1.* Canada: The Common Software Measurement International Consortium (COSMIC).

Albrecht, A. (1979). *Measuring AplicationDevelopment Productivity. Proc Of IBM applications. Development Joint.* Monterrey: Symposium.

Aranda, G. N. (2008). *Marco para la Elicitación de Requisitos Software en Procesos de Desarrollo Global.* España: Universidad de Castilla- La Mancha.

Arifoglu, A. (1993). Methodology for Software Cost Estimation. *ACM Sigsoft Software Engineering Notes 18(2)*, 96-105.

Aroba Páez, J. (2003). *Avances en la toma de decisiones en proyectos de desarrollo de software.* Sevilla: Universidad de Sevilla.

Bayley, J., & Basili, V. (1981). A meta-model for software development resource expenditures. *Proceedings of the 5th international conference on Software engineering.* San Diego, California, United States.

Benner, K., Feather, M., Johnson, W., & Zorman, L. (1993). *Utilizing Scenarios in the Software Development Process.* USA: Sciences Institute, Admiralty Way.

Bertolami, M. (2010). *Tesis Doctoral: ESTIMACIÓN DEL TAMAÑO FUNCIONAL DEL SOFTWARE EN LA ELICITACIÓN DE REQUERIMIENTOS.* Argentina: Facultad de Informática Universidad Nacional de La Plata.

Bloch, M. (01 de Octubre de 2012). *McKinsey & Company*. Obtenido de Delivering large-scale IT projects on time, on budget, and on value: http://www.mckinsey.com/insights/business_technology/delivering_large-scale_it_projects_on_time_on_budget_and_on_value

Bloch, M., Blumberg, S., & Laartz, J. (1 de Octubre de 2012). *Delivering large-scale IT projects on time, on budget, and on value*. Obtenido de http://www.mckinsey.com/insights/business_technology/delivering_large-scale_it_projects_on_time_on_budget_and_on_value

Boehm., B. (1981). *Software Engineering Economics.* New Jersey: Englewood Cliffs.

Chang, S. (2001). *Handbook of software engineering & knowledge engineering.* USA: World Scientific Publishing.

De la Torre, C., Zorrilla, U., Ramos, M. A., & Calvarro, N. (2010). *Guía de Arquitectura N-Capas orientada al dominio con .NET 4.0.* España: Krasis Consulting. S. L.

Debons, A., Horne, E., & Croneweth, S. (1988). *Information science: an integrated view.* Boston, EUA: G.K. Hall.

del Moral, A., Pazos, J., Rodríguez, F., Rodríguez-Patón, A., & Suárez, S. (2007). *Gestión del conocimiento.* Madrid: Paraninfo.

DeMarco, T. (1982). *Controlling Software Projects.* New Jersey: Yourdon Press.

Earl, M. (1998). *Knowledge as strategy: reflections on Skandia International and Shorko Films.* Boston, EUA: Butterworth-Heinemann.

Esterling, R. (1980). Software Manpower Costs. *Datamation*.

Gómez, J. (2014). *Guía Práctica de Estimación y Medición de Proyectos Software: ¿Por qué? ¿Para qué? y ¿Cómo?* Espaka: Safe Creative.

Gómez, J. (2014). *Guía Práctica de Estimación y Medición de Software.* España: SafeCreative.

Gonzáles Mateo, A. (2012). *Estimación Temprana mediante Puntos de Función IFPUG Tesis Magistral.* Alcalá de Henares: Universidad de Alcalá.

Gupta, J., & Sharma, S. (2004). *Creating knowledge-based organizations.* Harshey: Idea Group Inc.

Hadad, G. (2008). *Uso de Escenarios en la Derivación de Software, Tesis Doctoral.* La Plata, República Argentina: Facultad de Ciencias Exactas de la Universidad Nacional de La Plata.

Heemstra, F. (1992). Software Cost Estimation. *Information and Software Technology, Vol. 34*.

Helmer, O. (1967). *Analysis of the Future: The Delphi Method Santa Monica.* California: RandCorporation.

Humphrey, A., & Watts, S. (1995). *A Discipline for Software Engineering.* Massachusetts : Addison - Wesley.

IIBA_BABOKv2, I. I. (2009). *Guía sobre los fundamentos del Conocimiento del Análisis de Negocio (Guía BABOK) Versión 2.0.* Toronto, Ontario, Canadá: ©2005, 2006, 2008, 2009, International Institute of Business Analysis.

ISBSG. (27 de 03 de 2010). *International Software Benchmarking Standards Group*. Obtenido de http://www.isbsg.org/

Kendall, K., & Kendall, J. (2005). *Análisis y Diseño de Sistemas* (Sexta ed.). México, |: Pearson Educación.

Labdelaoui, H. (2001). *Métodos de Estimación de Tamaño Funcional del Software Aplicados a Enfoques de desarrollo.* NASSCOM India Leadership Forum (NILF). Inidia: NASSCOM.

Lai, J.-Y., Wang, C.-T., & Chow, C.-Y. (2008). How knowledge map and personalization affect effectiveness of KMS in high-tech firms. *International Conference on System Sciences* (pág. 355). Hawaii: Hawaii International Conference on System Sciences, Proceedings of the 41st Annual.

Liming, W. (4 de Marzo de 1997). *The Comparison of the Software Cost Estimating Methods.* Obtenido de http://www.compapp.dcu.ie/~renaat/ca421/LWu1.html

Longstreet, D. (2004). *Function Points Analysis Training Course.* Kansas: Longstreet Consulting Inc.

Louwers, R., Dekkers, T., & Timp, A. (2001). *FUNCTION POINT ANALYSIS FOR SOFTWARE ENHANCEMENT.* Netherlands: Professional guide of the Netherlands Software Metrics Users Association.

Marshall, C., Prusak, L., & Shpilberg, D. (1996). *Financial risk and the need for superior knowledge management.* Boston, EUA: Butterworth-Heinemann.

Matturro Mazoni, G. (2010). *Modelo para la gestión del conocimiento y la experiencia integrada a las prácticas y procesos de desarrollo software.* Madrid: Universidad Politécnica de Madrid.

Middleton, M. (1 de Mayo de 1999). *El profesional de la Información*. Obtenido de De la gestión de la información a la gestión del conocimiento: perspectivas sobre el desarrollo: http://www.elprofesionaldelainformacion.com/contenidos/1999/mayo/de_la_gestin_de_la_informacin_a_la_gestin_del_conocimiento_perspectivas_sobre_el_desarrollo.html

Peters, K. (1999). *Software Project Estimation.* California: .

Piirainen, K., Kivijarvi, H., & Touminen, M. (2008). Supporting strategic innovativeness: scenario planning for driving organizational knowledge sharing. *International Conference on System Sciences* (pág. 251). Hawaii: Hawaii International Conference on System Sciences, Proceedings of the 41st Annual.

PMI_PMBOKv5. (2013). *GUÍA DE LOS FUNDAMENTOS PARA LA DIRECCIÓN DE PROYECTOS (Guía del PMBOK®) — Quinta edición.* Newtown Square, Pensilvania 19073-3299 EE.UU.: Project Management Institute, Inc.

Polanyi, M. (1997). *The tacit dimension.* Boston, EUA: Butterworth-Heinemann.

Potts, C., & Antón, A. (1998). *A Representational Framework for Scenarios of System.* Atlanta: Technology, North Carolina State University, Engineering Graduate Research.

Pow Sang Portillo, J. (2012). *Técnicas para la estimación y planificación de proyectos de software con ciclos de vida incremental y paradigma orientado a objetos.* Madrid.

Rainardi, V. (2008). *Building a Data Warehouse.* Berkeley, CA 94705.: Apress@.

Sutcliffe, A. (1997). *Workshop Exploring Scenarios in Requirements Engineering.* Northampton Square: Centre for HCI Design, City University.

Symons, C. (1998). *MK II FUNCTION POINT ANALYSIS.* United Kingdom: United Kingdom Software Metrics Association.

Trafalis, T. (1999). Primal-dual optimization methods in neural networks and support vector machines training. *Thechnical report, School of industrial engineering*.

Weidenhaupt, K., Pohl, K., Jarke, M., & Haumer, P. (1998). *Scenario Usage in System Development: A Report on Current Practice.* Germany: RWTH Aachen, Informatik.

Wiig, K. (1993). *Knowledge management foundations.* Arlington: Schema Press.

Anexos

VII.1. ANEXO A:MATRIZ DE CONSISTENCIA

PROBLEMAS	OBJETIVOS	HIPOTESIS	VARIABLES	METODO
PROBLEMA PRINCIPAL	**OBJETIVO GENERAL**	**HIPOTESIS GENERAL**		**TIPO DE INVESTIGACION**
¿De qué manera el diseño y puesta en marcha de una metodología integral que permita la estimación funcional de proyectos de software, desde la elicitación de requisitos (BABOK), hasta la planificación del proyecto (PMBOK) complementada con las técnicas de estimación funcional (UFP, SMC/RICEF) y el léxico extendido del lenguaje (LEL), que sea retroalimentada por el descubrimiento de conocimiento de datos (KDD) de las casuísticas estructuradas presentadas en los proyectos post morten, puede mejorar la gestión de proyectos de software?	Establecer el grado de mejora de la gestión de proyectos de software al emplear una metodología integral que permita la estimación funcional de proyectos de software, desde la elicitación de requisitos (BABOK), hasta la planificación del proyecto (PMBOK) complementada con las técnicas de estimación funcional (UFP, SMC/RICEF) y el léxico extendido del lenguaje (LEL), que sea retroalimentada por el descubrimiento de conocimiento de datos (KDD) de las casuísticas estructuradas presentadas en los proyectos post morten.	El diseño y puesta en marcha de una metodología integral que permita la estimación funcional de proyectos de software, contribuye a mejorar la gestión de proyectos de *software*.	**VARIABLE INDEPENDIENTE** Estimación funcional **VARIABLES INTERVINIENTE** Metodología de estimación de proyectos basada en el descubrimiento de conocimiento de datos **VARIABLE DEPENDIENTE** Gestión de proyectos de software. **INSTRUMENTOS** Observación Estructurada Análisis de Variación Gestión del Valor Ganado (EVM) Minería de Datos(Clúster y Redes Neuronales) Pareto	Aplicada y correlacional **UNIVERSO:** Todas las empresas u organizaciones que desean realizar desarrollo de aplicaciones de Software. Especialmente las empresas dedicadas al desarrollo de Software. **UNIVERSO SOCIAL:** **Profesionales:** Ingenieros de Sistemas y carrera vinculadas al desarrollo de soluciones de software **Investigadores:** en temas de desarrollo de soluciones de software **MUESTRA:** Todos los proyectos y evolutivos de pequeño tamaño realizados por Edelnor desde el 2010 hasta el 2016 y STEFANINI – n CLIENTES desde el 2014 hasta el 2016.

PROBLEMAS PROBLEMAS SECUNDARIOS	OBJETIVOS OBJETIVO ESPECIFICOS	HIPOTESIS HIPOTESIS ESPECIFICA	*VARIABLES*	*INDICADORES*	*INDICES*	*METODO*
¿En qué medida la utilización de la estimación funcional sobre una metodología integral basada en el descubrimiento de conocimiento de datos, contribuye en la eficiencia de la gestión de proyectos de software en las organizaciones bajo estudio?	Establecer el grado de eficiencia en la gestión de proyectos de software, al utilizar la estimación funcional sobre una metodología integral de estimación, basada en el descubrimiento de conocimiento de datos	La utilización de la estimación funcional sobre una metodología integral de estimación, basada en el descubrimiento de conocimiento de datos contribuye en la eficiencia de la gestión de proyectos de software en las organizaciones bajo estudio.	**VAR. INDEPENDIENTE** Uso de la metodología integral de estimación. **VAR. DEPENDIENTE** Eficiencia en la gestión de Proyectos de software	▪ Puntos Función ▪ Tiempo ▪ Costo ▪ Eficiencia en la gestión de Proyectos	▪ Cantidad de PF de creación y actualización de cada proyecto por fase. ▪ Tiempo de duración de cada proyecto por fase. ▪ Costo de cada proyecto por fase. ▪ Variación del tiempo estimado con el real consumido (VTERC) por fase ▪ Índice del tiempo estimado con el real consumido del proyecto (ITERC) por fase. ▪ Variación del Cronograma (SV). ▪ Desempeño según cronograma (SPI). ▪ Variación del Costo (CV). ▪ Desempeño según Costo (CPI).	**TÉCNICAS** ▪ Observación estructurada ▪ Análisis de Variación ▪ Gestión del Valor Ganado (EVM)
¿En qué medida la utilización de la estimación funcional sobre una metodología integral basada en el descubrimiento de conocimiento de datos, contribuye en la eficacia de la gestión de proyectos de software en las organizaciones bajo estudio?	Establecer el grado de eficacia en la gestión de proyectos de software, al utilizar la estimación funcional sobre una metodología integral de estimación, basada en el descubrimiento de conocimiento de datos	La utilización de la estimación funcional sobre una metodología integral de estimación, basada en el descubrimiento de conocimiento de datos contribuye en la eficacia de la gestión de proyectos de software en las organizaciones bajo estudio.	**VAR. INDEPENDIENTE** Uso de la metodología integral de estimación. **VAR. DEPENDIENTE** Eficacia en la gestión de Proyectos de software	▪ Proyectos ▪ Eficacia en la gestión de Proyectos de software	▪ Cantidad de Proyectos por objetivo y fase ▪ Grado de cumplimiento del proyecto por objetivo y fase de estimación.	**TÉCNICAS** ▪ Pareto ▪ Análisis de Variación ▪ Análisis de Variación ▪ Minería de Datos
¿En qué medida la utilización de la estimación funcional sobre una metodología integral basada en el descubrimiento de conocimiento de datos, contribuye en la efectividad de la gestión de proyectos de software en las organizaciones bajo estudio?	Establecer el grado de efectividad en la gestión de proyectos de software, al utilizar la estimación funcional sobre una metodología integral de estimación, basada en el descubrimiento de conocimiento de datos	La utilización de la estimación funcional sobre una metodología integral de estimación, basada en el descubrimiento de conocimiento de datos contribuye en la efectividad de la gestión de proyectos de software en las organizaciones bajo estudio.	**VAR. INDEPENDIENTE** Uso de la metodología integral de estimación. **VAR. DEPENDIENTE** Efectividad en la gestión de Proyectos de software	▪ Puntos Función ▪ Tiempo ▪ Efectividad en la gestión de Proyectos de software	▪ Cantidad de PF por tipo de proyecto y tecnología empleada ▪ Cantidad de Horas por tipo de proyecto y tecnología empleada ▪ Productividad del equipo del proyecto por tecnología empleada.	**TÉCNICAS** ▪ Observación estructurada ▪ Análisis de Variación

VII.2. ANEXO B: DEFINICIÓN DE TERMINOS

AC COSTO ACTUAL. Representa el monto que gastamos para completar el trabajo.

***AGILE*.** Iteraciones de desarrollo de aplicaciones de *software* de estilos 'ágiles', cortos y enfocados. Estos proyectos puede que no requieran ninguna o muy poca documentación formal de los requerimientos. Pizarras, rotafolios e historias de usuarios pueden ser suficientes. El estilo 'ágil' se enfoca en crear un mínimo necesario de documentación para entregar los requerimientos, y muchos equipos 'ágiles' preferirán documentar la solución después de que ha sido entregada.

APO ACTIVOS DE LOS PROCESOS DE LA ORGANIZACIÓN. Los activos de los procesos de la organización son los planes, los procesos, las políticas, los procedimientos y las bases de conocimiento específicos de la organización ejecutora y utilizados por la misma. Estos incluyen cualquier objeto, práctica o conocimiento de alguna o de todas las organizaciones que participan en el proyecto y que pueden usarse para ejecutar o gobernar el proyecto. Los activos de procesos también incluyen bases de conocimiento de la organización como lecciones aprendidas e información histórica. Los activos de los procesos de la organización pueden incluir cronogramas completados, datos sobre riesgos y datos sobre el valor ganado. Los activos de los procesos de la organización pueden agruparse en dos categorías: (1) procesos y procedimientos, y (2) base de conocimiento corporativa.

BA *BUSINESS ANALYST*. Esta persona debe estar próximo al usuario final y entender los problemas que tiene en el día a día para desarrollar su trabajo y al mismo tiempo debe conocer cómo están construidas las aplicaciones, incluso haber tenido experiencia en el desarrollo de las mismas. De esta forma se convierte en un usuario técnicamente avanzado, por lo que puede ayudar a definir soluciones que cubran las necesidades del negocio con el menor impacto posible en las aplicaciones existentes. El conjunto de tareas y técnicas que se utilizan para llevar a cabo el análisis de negocio se definen en la guía de los fundamentos de la *Business Analysis of Knowledge* ® (Guía *BABOK* ®*)*

BD. Almacén o repositorio de datos de los proyectos desde la etapa de elicitación hasta el cierre.

BI. Desde un punto de vista pragmático, y asociándolo directamente con las tecnologías de la información, podemos definir *Business Intelligence* como el conjunto de metodologías, aplicaciones y tecnologías que permiten reunir, depurar y transformar datos de los sistemas transaccionales e información desestructurada (interna y externa de la organización) en información estructurada, para su explotación directa (reporting, análisis OLTP / OLAP, alertas) o para su análisis y conversión en conocimiento, dando así soporte a la toma de decisiones sobre el negocio.

BOTTON-UP. Esta Metodología consiste en reunir diferentes sistemas que conforman un todo. Los elementos individuales son especificados en gran detalle, los componentes se van uniendo unos con otros hasta conformar el sistema final, que se logra al llegar al nivel superior. Esta estrategia asemeja al modelo "semilla", en el cual se parte de algo pequeño que va creciendo hasta llegar a un sistema terminado y complejo

BUSINESS CASE **(CASO DE NEGOCIO).** Evaluación de los costos y beneficios asociados a una iniciativa propuesta.

BUSINESS NEED **(NECESIDAD(ES) DE NEGOCIO).** Tipo de requerimiento de negocio de alto nivel que es establecido como un objetivo de negocio, o un impacto que la solución debe tener en su ambiente.

CLUSTER. Algoritmo que intenta agrupar una serie de objetos en grupos. Cada objeto es representado por un vector de atributos n-dimensional. Los objetos que forman cada grupo deben ser disimilares además la similaridad es medida del grado de proximidad.

CMMI (INTEGRACIÓN DE MODELOS DE MADUREZ DE CAPACIDADES). Modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software

COCOMO (*CONSTRUCTIVE COST MODEL*). Modelo paramétrico usado para estimar el esfuerzo y calendario de un proyecto de desarrollo de *software*.

CPI. Desempeño según presupuesto

***CRASHING* (INTENSIFICACIÓN).** Una técnica utilizada para acortar la duración del cronograma con el menor incremento de costo mediante la suma de recursos.

CSCWS SISTEMAS DE TRABAJO COLABORATIVO APOYADOS POR COMPUTADORA (*COMPUTER SUPPORTED COLLABORATIVE WORK SYSTEMS*). Pueden contener un respaldo de un tipo de *software* denominado groupware para la colaboración en equipo a través de computadoras conectadas en red.

DSS *DECISION SUPPORT SYSTEMS*. sistema informático utilizado para servir de apoyo, más que automatizar, el proceso de toma de decisiones

EDT/WBS ESTRUCTURA DE DESGLOSE DEL TRABAJO (*WORK BREAKDOWN STRUCTURE*). Estructura jerárquica formada por el conjunto de tareas (paquetes de trabajo) y entregables necesarios para completar un proyecto.

EI (*EXTERNAL INPUTS*). Entradas externas.

ELF (*EXTERNAL LOGICAL FILE*). Archivos lógicos externos

EO (*EXTERNAL OUTPUT*). Salidas externas

EQ (*EXTERNAL QUERIES*). Consultas externas

ESS *EXECUTIVE SUPPORT SYSTEMS*. sistema informático utilizado para ayudar a los ejecutivos a organizar sus actividades relacionadas con el entorno externo mediante herramientas gráficas y de comunicaciones. Algunas aplicaciones se originan para la ayuda al comercio electrónico.

***EXECUTING PROCESSES* (PROCESOS DE EJECUCIÓN).** Aquellos procesos realizados para completar el trabajo definido en el plan para la dirección del proyecto a fin de satisfacer las especificaciones del mismo.

EV VALOR GANADO. Es una medida del valor del trabajo que se completó a un momento determinado.

***FAST TRACKING* (EJECUCIÓN RÁPIDA).** Una técnica de compresión del cronograma en la que actividades o fases que normalmente se realizan en secuencia se llevan a cabo en paralelo por menos durante una parte de su duración.

FPA *(ADJUSTED FUNCTION POINTS).* Puntos de función empleando los factores de ajuste no reconocidas en el ISO.

FRAMEWORK. desde un punto de vista técnico es un esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación

GDSS *COMPETER SUPPORTED COLLABORATIVE WORK SYSTEMS.* Sistema interactivo basado en computadora, el cual facilita la solución de problemas no estructurados por un conjunto de tomadores de decisiones trabajando juntos como un grupo.

GI. Gestión de la información.

GRH. Gestión de recursos humanos

IEEE (*THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS*). Asociación técnico- profesional dedicada a la estandarización en el campo de la tecnología, también se encarga de la promover la creatividad, el avance e integración de los avances tecnológicos.

IFPUG. es el *International Function Points Users Groups* o, en español, el Grupo Internacional de Usuarios de Puntos Función.

ILF (*INTERNAL LOGICAL FILE*). Archivos lógicos internos

***INITIATING PROCESSES* (PROCESOS DE INICIO).** Aquellos procesos realizados para definir un nuevo proyecto o nueva fase de un proyecto existente al obtener la autorización para iniciar el proyecto o fase.

ISACA ASOCIACIÓN DE AUDITORÍA Y CONTROL DE SISTEMAS DE INFORMACIÓN *(INFORMATION SYSTEMS AUDIT AND CONTROL ASSOCIATION).* Una asociación internacional que apoya y patrocina el desarrollo de metodologías y certificaciones para la realización de actividades auditoría y control en sistemas de información.

ISO ORGANIZACIÓN INTERNACIONAL DE NORMALIZACIÓN. Es la organización que se ocupa de establecer las normas de fabricación, de comunicación y de comercialización, tanto de productos como de servicios, en el plano internacional. Lo que básicamente se propone el ISO es estandarizar las normas

ITERC. Índice de optimización del proyecto.

ITIL. Biblioteca de infraestructura de tecnologías de información.

KDD *KNOWLEDGE DISCOVERY FROM DATABASES.* Se refiere al proceso no-trivial de descubrir conocimiento e información potencialmente útil dentro de los datos contenidos en algún repositorio de información. No es un proceso automático, es un proceso iterativo que exhaustivamente explora volúmenes muy grandes de datos para determinar relaciones.

LÍNEA BASE. Especificación o producto que se ha revisado formalmente y sobre el cual se ha llegado a un acuerdo, y que de ahí en adelante sirve como base para un desarrollo posterior.

LLUVIA DE IDEAS. Es una herramienta de trabajo grupal que ayuda el surgimiento de nuevas ideas sobre un tema o problema determinado, la técnica se basa en una reunión en donde los participantes generan ideas sobre el tema tratado.

LOC. Líneas de código.

NEURAL NETWORK. Son un paradigma de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso de los animales. Se trata de un sistema de interconexión de neuronas que colaboran entre sí para producir un estímulo de salida. En inteligencia artificial es frecuente referirse a ellas como redes de neuronas o redes neuronales.

OLAP (*ON-LINE ANALYTICAL PROCESSING*). Es una solución utilizada en el campo de la llamada Inteligencia de negocios (o *Business Intelligence*) cuyo objetivo es agilizar la consulta de grandes cantidades de datos

OLTP (*ONLINE TRANSACTION PROCESSING*). Tipos de programa que facilita y gestiona aplicaciones orientadas a transacciones, típicamente para ingreso de datos y retención de transacciones en varias industrias, incluyendo banca, aerolíneas, correo, supermercados y productores.

PADDING. Establece el tiempo de relleno asociado a una actividad.

PDS. Proyectos de Desarrollo de Software

PIB PRODUCTO INTERIOR BRUTO. Conjunto de los bienes y servicios producidos en un país durante un espacio de tiempo, generalmente un año.

***PLANNING PROCESSES* (PROCESOS DE PLANIFICACIÓN).** Aquellos procesos requeridos para establecer el alcance del proyecto, refinar los objetivos y definir el curso de acción requerido para alcanzar los objetivos propuestos del proyecto.

PMBOK. Es una colección de procesos y áreas de conocimiento generalmente aceptadas como las mejores prácticas dentro de la gestión de proyectos. Este estándar fue construido por el *Project management institute*

POST MORTEM. Proceso de evaluación del proyecto, que verifica las metas establecidas en el plan de calidad.

PUNTO DE FUNCIÓN (*FUNCTION POINT*). Medida del tamaño de un sistema de *software* y del proyecto que lo construye, esta medida se basa en la teoría de que la funcionalidad del software es la mejor medida de su tamaño.

PV VALOR PLANIFICADO. Representa el costo planificado del trabajo que debería estar completo en un momento determinado.

REQUERIMIENTOS FUNCIONALES. Estos son las funciones que el sistema en desarrollo será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas.

ROM (*ROUGH ORDER OF MAGNITUDE).* Es la estimación menos precisa que se utiliza al principio del proyecto, cuando la información es limitada, y por lo tanto menos precisa. La Guía del PMBOK quinta edición da las pautas de los niveles de precisión en -25 % a + 75 %, o potencialmente aún mayor.

SEI (*SOFTWARE ENGINEERING INSTITUTE*). Instituto federal de investigación, dedicado a la investigación de temas relacionados con la ingeniería de software y el mejoramiento en el proceso de desarrollo de software.

SPI. Desempeño según cronograma.

SRS (*SOFTWARE REQUERIMENTS SPECIFICATION*). Documento donde se definen de forma precisa los requerimientos funcionales del software que se va a construir.

***STAKEHOLDERS*.** Una persona o grupo de personas que tienen intereses que pueden verse afectados por una iniciativa o influir en ella.

TI. Es el estudio, diseño, desarrollo, implementación, soporte y administración de los sistemas de información basados en computadoras, particularmente aplicaciones de *software* y *hardware* de computadoras.

TOP-DOWN. Se formula un resumen de la presupuestación, sin especificar detalles. Cada parte del presupuesto se refina diseñando con mayor detalle. Cada

parte nueva es entonces redefinida cada vez con mayor detalle, hasta que la especificación del presupuesto este lo suficientemente detallado.

VTERC. Variación del tiempo estimado con el real consumido

WATERFALL (MODELO EN CASCADA). (Denominado así por la posición de las fases en el desarrollo de esta, que parecen caer en cascada "por gravedad" hacia las siguientes fases), es el enfoque metodológico que ordena rigurosamente las etapas del proceso para el desarrollo de *software*, de tal forma que el inicio de cada etapa debe esperar al término de la etapa anterior. Al final de cada etapa, el modelo está diseñado para llevar a cabo una revisión final, que se encarga de determinar si el proyecto está listo para avanzar a la siguiente fase. Este modelo fue el primero en originarse y es la base de todos los demás modelos de ciclo de vida.

WHITE PAPER **(LIBRO BLANCO).** Es un documento o guía con autoridad con el objetivo de ayudar a los lectores a comprender un tema, resolver o afrontar un problema (por ejemplo diseñando standares de desarrollo de *software*), o tomar una decisión.

VII.3. ANEXO C: MODELO DE ESCENARIOS LEL

Escenario: descripción de una situación en el dominio de la aplicación.
Sintaxis: Titulo + Objetivo + Contexto + $\{Recursos\}_1^N$ + $\{Actores\}_1^N$ + $\{Episodios\}_2^N$ + {Excepciones}

Titulo: identificación del escenario. En el caso de un subescenario, el título es el mismo que la sentencia del episodio, sin las restricciones.
Sintaxis: Frase | ([Actor | Recurso] + Verbo + Predicado)

Objetivo: meta a ser alcanzada en el dominio de la aplicación. El escenario describe el logro del objetivo.
Sintaxis: [Actor | Recurso] + Verbo + Predicado

Contexto: compuesto al menos por uno de los siguientes subcomponentes:
Ubicación Geográfica: ubicación física del escenario.
Ubicación Temporal: especificación de tiempo para el desarrollo del escenario.
Precondición: estado inicial del escenario.
Sintaxis: {Ubicación Geográfica} + {Ubicación Temporal} + {Precondición}
donde Ubicación Geográfica es:
Frase + {Restricción}
donde Ubicación Temporal es:
Frase + {Restricción}
donde Precondición es:
[Sujeto | Actor | Recurso] + Verbo + Predicado + {Restricción}

Recursos: elementos físicos relevantes o información que debe estar disponible en el escenario.
Sintaxis: Nombre + {Restricción}

Actores: personas, dispositivos u organización que tienen un rol en el escenario.
Sintaxis: Nombre

Episodios: conjunto de acciones que detallan el escenario y proveen su comportamiento. Un episodio también puede ser descripto como un escenario.
Sintaxis (usando BNF parcial):
<episodios> ::= <serie de grupos> | <serie de episodios>
<serie de grupos> ::= <grupo> <grupo> | <grupo no secuencial> | <serie de grupos> <grupo>
<grupo> ::= <grupo secuencial> | <grupo no secuencial>
<grupo secuencial> ::= <sentencia básica> | <grupo secuencial> <sentencia básica >
<grupo no secuencial> ::= # <serie de episodios> #
<serie de episodios > ::= <sentencia básica > <sentencia básica> |
<serie de episodios> <sentencia básica >
<sentencia básica> ::= <sentencia simple> | <sentencia condicional> | <sentencia opcional>
<sentencia simple> ::= <sentencia episodio> CR
<sentencia condicional> ::= **IF** <condición> **THEN** <sentencia episodio> CR
<sentencia opcional> ::= **[** <sentencia episodio > **]** CR
donde <sentencia episodio > es descripta:
(([Actor | Recurso] + Verbo + Predicado) | ([Actor | Recurso] + [Verbo] + Título)) + {Restricción}

Excepciones: usualmente reflejan la falta o mal funcionamiento de un recurso necesario. Una excepción impide el cumplimiento del objetivo del escenario. El tratamiento de la excepción puede ser expresado mediante otro escenario.
Sintaxis: Causa [(Solución)]
donde Causa es:
Frase | ([Sujeto | Actor | Recurso] + Verbo + Predicado)
donde Solución es:
Titulo

Restricción: un requerimiento de alcance o calidad referido a una entidad dada. Es un atributo de los Recursos, Episodios básicos o subcomponentes de Contexto.
Sintaxis: ([Subject | Actor | Resource] + **[No] Debe** + Verbo + Predicado) | Frase

+ significa composición, {x} significa cero o más ocurrencias de x, ( ) es usado para agrupar, | significa **or** y [x] denota que x es opcional.

Imagen 20: Modelo de Escenarios. Fuente: (Bertolami, 2010)

VII.4. ANEXO D: PLANTILLA PARA LA TOMA DE DATOS DE PUNTOS FUNCIÓN.

PlantillaTomaDatosPuntosFuncion

Plantilla para toma de datos