Instituto Politécnico Nacional

Centro de Investigación en Computación

Secretaría de Investigación y Posgrado

"Prototipo de una herramienta de apoyo para la estimación de costos de un proyecto de software"

T E S I S

QUE PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A

EL LICENCIADO EN INFORMÁTICA:

ANTONIO RAMÍREZ RAMÍREZ

DIRECTOR DE TESIS: **M. en C. SANDRA DINORA ORANTES JIMÉNEZ**

MÉXICO, D.F. **JUNIO 2011**

SIP-14

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 08:00 horas del día 29 del mes de Junio de 2011 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"PROTOTIPO DE UNA HERRAMIENTA DE APOYO PARA LA ESTIMACIÓN DE COSTOS DE UN PROYECTO DE SOFTWARE"

Presentada por el alumno:

RAMÍREZ	**RAMÍREZ**	**ANTONIO**
Apellido paterno	Apellido materno	Nombre(s)

Con registro: **B 0 9 1 6 5 3**

aspirante de: **MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Director de Tesis

M. en C. Sandra Dinora Orantes Jiménez

Dr. Sergio Suárez Guerra

Dr. Marco Antonio Moreno Armendáriz

Dr. Marco Antonio Moreno Ibarra

Dr. Miguel Jesús Torres Ruiz

PRESIDENTE DEL COLEGIO DE PROFESORES

Dr. Luis Alfonso Villa Vargas

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION
EN COMPUTACION
DIRECCION

RESUMEN

El presente documento se enfoca en un tema importante para la Ingeniería de Software, como lo es la e*stimación de costos en proyectos de software*. Se realiza un análisis de los modelos de costos existentes, en sus versiones más recientes así como los que son considerados métricas de la Ingeniería de Sistemas Computacionales. Dos de las métricas más utilizadas son el COCOMO II (COnstructive COst MOdel, Modelo Constructivo de Costos) e IFPUG (International Function Point Users Group, Grupo Internacional de Usuarios de Puntos Función) conocido este último como Métrica por Puntos de Función.

La viabilidad de aplicar estimación de costos al desarrollo y creación de software, se ha convertido en cierta medida en una prioridad para los ingenieros de desarrollo. Dicha estimación será de gran utilidad en la toma de decisiones, siempre que el proyecto en cuestión, se apegue a los lineamientos que fija la propia Ingeniería de Software, área que es conciliada y reconocida por organismos como la IEC/ISO (International Electro technical Commission / International Organization for Standardization).

Se presenta a colación, una herramienta de software que automatiza la aplicación de las dos métricas mencionadas y propone una combinación éstas, con la idea de obtener una tercera opción de metodología para obtener la estimación de costos de un proyecto de software, manteniendo un enfoque más práctico que teórico.

En el presente trabajo se presenta también un análisis de los antecedentes a dichas métricas, de su utilidad y forma de aplicación, esbozando además una comparación entre los resultados de cada una de ellas contra los resultados de la opción propuesta que es una combinación de ambas; de donde se desprende la utilidad de utilizar la herramienta de software diseñada.

ABSTRACT

This document, get focus on very important issue from Software Engineering as is Costs Estimation of software projects. It's make an analysis , on intent to presents the proposal and development of a software tool that provides support in the evaluation and costs estimation as far as time and resources talk about, of a software project, with base in the most recent models of measurement on the Engineering of Computing Systems; such as COCOMO (constructive cost model) and IFPUG (international function point users group) known this last one like function points software measurement; whenever the project at issue, is become attached under rules that the own Software Engineering fixes, that is restrained by organisms like the ICE/ISO (International Electrotechnical Commission/International Organization for Standardization) to guarantee so much to the developer as to the user, which the project is made keeping certain guarantees of efficiency and quality.

CONTENIDO

Índice de Tablas.

Índice de Figuras:

Capítulo I

INTRODUCCION

Competencia, es un término utilizado tanto en la industria internacional, como en las grandes empresas de servicios, pero las entidades mexicanas de cualquier rubro, muchas veces no están a la altura de las del resto del mundo. En el área de desarrollo de software sucede algo similar pero no por falta de ingenio o calidad en los algoritmos solución, sino en la poca seriedad con que se trata o aplica la Administración de Proyectos [2].

En México y en algunos países de América Latina por lo general la creación y desarrollo de software está a cargo de ingenieros en sistemas, programadores y a veces de ingenieros en computación, pero realmente no se da al Ingeniero de Software la importancia que tiene. Por lo tanto, la industria del software en estos países, comúnmente no se interesa en las cualidades y alcances de los proyectos, en ocasiones ni siquiera se tiene idea de los mecanismos de evaluación de productos de software que existen en otras partes del mundo.

Sin embargo los desarrolladores y las empresas de desarrollo de software que sí conocen y se ajustan a los estándares y lineamientos de la administración de proyectos de software, entienden la importancia de aplicar los modelos de evaluación que existen y tratan de seguir rutas adecuadas en el desarrollo de Software. Cuando estas empresas emprenden desarrollos importantes, es porque conocen la importancia de comprometerse con la correcta aplicación de un ciclo de evaluación constante, que les permita establecer niveles de eficiencia capaces de llevarlos al cumplimiento de contratos [6].

Se ha demostrado que uno de los principales problemas de las empresas de servicios en casi toda América Latina, es el incumplimiento de los contratos, la mayoría de estos incumplimientos son en tiempos de entrega, más que en contenido. Con lo anterior es posible deducir, que los problemas internos que padecen, no son en cuanto a capacidad o desempeño, sino más bien en la etapa de planeación [8].

Las métricas de software se refieren a una gama de metodologías para la medición de la calidad del producto, dentro del contexto de la planificación de los proyectos de software, estas métricas de calidad pueden ser aplicadas tanto a las organizaciones y sus procesos, como al personal que participa y los productos que desarrollan; las cuales afectan directamente a la estimación de costos [3].

Dichas metodologías podrían clasificarse en: Métricas de productividad, son las que se centran en el rendimiento del proceso de la ingeniería de software; Métricas de Calidad, que proporcionan una indicación de cómo se ajusta el software, a los requerimientos implícitos y explícitos del cliente; Métricas Técnicas, se centran en el carácter y esencia del software, más que en el proceso de su desarrollo; Métricas Orientadas al Tamaño, que son utilizadas para obtener medidas directas del resultado y su calidad de ingeniería en su aplicación; Métricas Orientadas a la Función, que son medidas indirectas del software como producto y de su proceso de creación, centrándose en su funcionalidad o utilidad y por último las Métricas Orientadas a la persona, que consiguen información sobre la forma en que la gente desarrolla software de computadora y sobre el punto de vista humano de la efectividad de las herramientas y métodos [2].

Toca el turno a la Ingeniería de Software, entender que al aplicar la Administración de Proyectos, solo será efectiva si es capaz de implementar las métricas de evaluación en sus procesos de "mejora continua". Con el presente trabajo, se pretende participar directamente en dicha mejora, proponiendo una alternativa a las metodologías existentes, que surge de la combinación de algunas características de las métricas que fueron analizadas y que por su grado de eficacia son las más utilizadas. Dicha alternativa está confeccionada en una herramienta capaz de auxiliar al Ingeniero de Software.

Sin embargo ante el hermetismo de algunas empresas u organismos que aplican dichas métricas ha sido necesaria una profunda investigación para determinar el modelo de aplicación y la posibilidad de automatizarlas. Se ilustra entonces cómo esta herramienta se plantea desde el punto de vista de la Ingeniería de Software, con lo que el término en sí representa: una manera ordenada, probada y establecida de generar aplicaciones de cómputo [5].

1.1 ANTECEDENTES.

Ingeniería del Software

Puede definirse a la Ingeniería de Software a partir de los términos Ingeniería, que es la aplicación de la ciencia y las matemáticas para que las propiedades de la materia y las fuentes de energía beneficien al hombre y Software que son el conjunto de algoritmos, programas, procedimientos y recursos técnicos involucrados en un determinado sistema de información automatizado. La Ingeniería de Software es entonces la aplicación de la ciencia y las matemáticas para que las bondades de un sistema de Información sean útiles al hombre mediante la solución de determinados problemas [1].

En este orden de ideas, la correcta aplicación de la Ingeniería de Software en un determinado problema, representa la correcta realización de un software de Computadora, es decir la planeación, diseño y desarrollo de una solución que permita obtener una herramienta cómoda e intuitiva que ayude a sortear algún obstáculo.

Ante ello, han surgido métodos y metodologías orientadas a evaluar tanto el desarrollo de dichas aplicaciones como los aspectos relacionados con las mismas, como lo son la correcta planeación, la organización de su diseño y desarrollo y hasta el tiempo que tardan en ser desarrolladas así como la calidad del software logrado.

Algunos de estos modelos y estándares, se han aplicado desde finales del siglo pasado en las grandes empresas de desarrollo de software y por lo tanto han evolucionado. De tal forma que aunque hoy existen varias métricas para el desarrollo y la calidad de los proyectos de software, se han convertido en las mejores al punto de ser consideradas estándares, por un lado el modelo de costos COCOMO (Constructive Cost Model) [3]. La palabra "constructive" se refiere a el hecho que el modelo ayuda a un estimador a comprender mejor la complejidad del software; este modelo es un ejemplo de variable simple estático y hoy en día es usado por miles de administradores de proyecto de Software.

COCOMO ayuda a estimar el esfuerzo, tiempo, gente y costos (ya sea estos de desarrollo, equipamiento y mantenimiento), que provee tres "niveles" de aplicación, basados en los factores considerados por el modelo. Básico, con el enfoque de un modelo estático simplemente evaluado que calcula el esfuerzo (y costo) del desarrollo del software como función del programa expresado en líneas de código (LDC estimados)[7]; Intermedio, que calcula el esfuerzo del desarrollo del software como su función, así como

el tamaño del programa y un conjunto de "guías de costo" que incluye una evaluación subjetiva del producto, hardware, personal y de los atributos del proyecto y por último, un nivel avanzado, que incorpora todas las características de la versión intermedia con una evaluación del impacto de las vías de costo en cada fase (análisis, diseño, etc.) del proceso de la ingeniería de software [12].

Por otro lado la famosa métrica de puntos función, avalada por el organismo IFPUG (International Function Point Users' Group) [17], quienes se anuncian como una organización no lucrativo (pero que vende su método), con la misión de ser un líder reconocido en promover y animar la gerencia eficaz de las actividades del desarrollo y del mantenimiento del software de usuario, mediante el uso del análisis del punto de la función y de otras técnicas de la medida del software.

1.2 Planteamiento del Problema.

Existen diferentes metodologías o estándares de medición, una de las más populares es la mantenida por el International Function Point Users Group (IFPUG) [17]. La métrica del punto función es un método utilizado en Ingeniería de Software para medir el tamaño del software. Fue definida por Allan Albrecht, de IBM, en 1979 [10] y pretende medir la funcionalidad entregada al usuario independientemente de la tecnología utilizada para la construcción y explotación del software. También propone ser de utilidad para cualquiera de las fases del proceso de vida del software.

Ésta, como algunas otras metodologías se emplean en la industria de software, reportando todas ellas un grado de éxito, sin embargo, no es posible establecer un parámetro de medición de resultados de dichas métricas en empresas de habla hispana en México y en la mayor parte América latina, ya que no son ampliamente conocidas y por ende no se aplican; en algunos casos aunque se conozca de ellas, resulta un tanto difícil establecer o implementar una nueva norma, cuando se carece de la cultura empresarial e informática necesarias.

Apoyar la aplicación o implementación de alguna de estas metodologías, con la creación de una herramienta sencilla y de fácil acceso, podría ayudar a que un porcentaje de las empresas de desarrollo de software, intenten mecanismos de autoevaluación y de apoyo a la toma de decisiones dentro de sus organismos, para hacer eficiente la contratación de nuevos proyectos.

1.3 Objetivos.

1.3.1 Objetivo General.

Presentar un prototipo de herramienta de software, que sirva de apoyo para la estimación de costos en la etapa de diseño de un proyecto de software cualquiera, que facilite el uso o aplicación de al menos una de las metodologías mayormente utilizadas y probadas en esta área.

1.3.2 Objetivos Específicos.

Ahondar en la indagación de los trabajos reportados que utilicen las métricas IFPUG y COCOMO, para establecer un correcto estado del arte.

Comparar entre éstos métodos de evaluación e intentar fusionarlos como parte de una herramienta de software que automatice sus funciones

Iniciar el Diseño de la aplicación, siguiendo las normas de la Ingeniería de Software, para estar en posibilidad de demostrar la utilidad de las mismas.

1.4 Justificación.

La razón fundamental por la que la Estimación de Costos en la producción de software resulta de gran importancia, es porque en la práctica, se carece enormemente de metodologías confiables para la correcta administración de proyectos de software. Por un lado prevalece la necedad de los ingenieros en computación que insisten en diseñar y construir programas de computadora, sin llevar a cabo una planificación adecuada que implique la realización de un contrato de servicios o cuando éste existe, la mayoría de las veces es de manera verbal. Ellos, comúnmente emprenden el desarrollo de software sin una metodología a seguir, provocando con ello que el mercado de software local en muchos países de América Latina, incluyendo a México, sea prácticamente nulo, comparándolo con el desarrollo y distribución de software en países industrializados. Por otro lado, tenemos que la unión entre los conceptos generales del análisis económico y el mundo de la Ingeniería de Software no es cosa sencilla, pero es preciso entender que no hay forma de realizar un análisis costo—beneficio sobre el Desarrollo de Software, si no se cuenta con un método de Estimación de Costos para evaluar o presupuestar dichos desarrollos, así como contar con las herramientas necesarias para el análisis de la influencia que ejercen los demás factores que intervienen en la creación del producto, ya sea del proyecto en sí o del entorno de la producción del mismo.

De este modo las técnicas de estimación son importantes porque proporcionan la parte esencial de una buena gestión de proyectos, ya que sin una aceptable capacidad de estimación de software, los proyectos podrían estar sufriendo todos o alguno de los siguientes problemas:

1. Los programadores no tienen una base firme sobre la que apoyarse para afirmar a su cliente o patrón, que el tiempo y los recursos que le han sido otorgados para finalizar el producto son suficientes o en su caso, poco realistas.

2. Los analistas de software en México, comúnmente no tienen forma de realizar un análisis fiable de intercambio de piezas hardware—software durante la fase de diseño del sistema. Esto puede provocar un diseño de software en el que el desempeño del hardware, quede por debajo de lo esperado a causa de un software que ha costado incluso, mucho más de lo estimado.

3. Los gestores del proyecto no saben cómo estimar el tiempo y el esfuerzo que conlleva cada fase y actividad durante el desarrollo de un determinado proyecto.

1.5 Beneficios Esperados

Tradicionalmente se ha presentado que según la forma en que es llevada a cabo la Ingeniería de Software, es como se determina el coste y la calidad del software producido. Esto hace precisamente a la Ingeniería de Software importante por los siguientes aspectos:

1. El software cada vez requiere ser más específico.
2. El software es cada vez más costoso.
3. El software precisa de ser liberado en menos tiempo, so pena de caer en obsolescencia.
4. El software tiene cada vez más un impacto importante en el bienestar del ser humano.

Actualmente, puede considerarse al hardware como la “caja negra” que contiene al software, siendo el mismo hardware el que determina el costo de un Sistema de Información. Por otro lado, el incremento en la demanda de programas de computadora que sean cada vez más fácil de usar, es un gran desafío para la Ingeniería de Software, primero, porque aumenta significativamente la productividad en el desarrollo de software, segundo, porque incrementa la necesidad y eficacia del software de mantenimiento.

Este incremento en la demanda de software resulta mucho más pronunciado por el hecho de que el hardware cada vez es más barato y confiable y es aquí donde entra la labor de la Estimación de Proyectos Informáticos. Poder acotar de forma notable el costo de producción para minimizar riesgos y represente una ventaja para la organización o, en caso contrario, prevenir a la organización de un desarrollo que puede acarrearle un retroceso, e incluso la ruina; será un logro esperado. Por todo ello, se hace verdaderamente útil contar con una herramienta que coadyuve en una de las más difíciles tareas que enfrenta el Ingeniero de Software que es la de prever el nivel de gastos que enfrentará al realizar un desarrollo nuevo.

1.6 Alcance y Límites

Es importante hacer una distinción entre una metodología de Administración de Proyectos y una de Desarrollo de Software ya que debe establecerse que ambos conceptos aunque pertenecen a un mismo tema y se encuentran relacionados dentro de una organización, ésta debe aplicar de forma clara ambos y contar con una metodología de administración de proyectos que sea consistente con cualquiera que sea la naturaleza del proyecto que desarrolla. Por otra parte debe contarse también con una metodología que esté relacionada específicamente con el desarrollo de un proyecto de software, ya que en la vida real casi no sucede, aún tratándose en ocasiones de una organización dedicada a la industria del software [4].

Uno de los puntos principales de la Metodología de Desarrollo de Aplicaciones es aquella que propone que los proyectos deben ser divididos en fases y que antes de iniciar con cada una de esas etapas debe existir un plan que las establezca, qué defina las actividades que involucran, así como los roles y responsabilidades de cada elemento involucrado [5]; para así pasar a la fase de Presupuestar cada nuevo proyecto.

En esta fase o etapa de la Estimación es en donde interviene o en donde servirá de apoyo una herramienta de software, que no solo sea más precisa sino mucho más rápida además de ofrecer la posibilidad de probar con diferentes opciones, para dar oportunidad al usuario de alternar con las variables de que dispone y plantarse a sí mismo y ante la junta directiva diferentes escenarios y proyecciones de un mismo desarrollo [1].

En este sentido, la herramienta software que se propone deberá proporcionar la suficiente utilidad al Ingeniero de Software, sin importar el giro o ámbito en el que se desenvuelva. Deberá ser capaz de ofrecer una ventaja significativa, sobre cualquier método que esté empleando actualmente para realizar las actividades que le permiten conocer de manera anticipada y tal vez solo aproximada, del nivel de gastos que enfrentara al emprender un proyecto de software. Al tener conocimiento de dicha información podrá anticiparse a cualquier eventualidad o inclusive tomar la decisión de llevar al cabo o no, aquel proyecto.

La utilidad adicional o ventaja que el Ingeniero de Software adquirirá al utilizar la herramienta propuesta, será la de contar en un mismo instrumento, la evaluación y apoyo que brindan dos métricas por separado que son el COCOMOII [15] y los FunctionPoints [14]. Como se ha comentado en el apartado de

los Antecedentes y como se verán a detalle en la fase del marco teórico, estas métricas que fueron propuestas y demostradas en los ochenta, se han utilizado desde entonces en el ámbito comercial con éxito.

La idea principal es que el Ingeniero de Software, pueda tener acceso a la aplicación o utilización de estas métricas de software que principalmente auxilian en la etapa de planeación del software, desde una misma herramienta desarrollada en español y con la suficiente ayuda en pantalla para que realmente sea un apoyo y no un obstáculo su utilización.

Por otro lado habrá de comprender antes el Ingeniero de Software, que no será posible aplicar los mismos métodos para cada caso de estudio que se presenta, pues como lo habrá experimentado ya, en cada situación se encuentran requerimientos diferentes y dependiendo de las veces que haya estado sujeto a resolver situaciones similares, se habrá vuelto experto en mayor o menor medida a resolver mejor algún tipo de problema más que otro, por ello se hace hincapié en que no deberá esperarse que una herramienta de este tipo, sea la solución a toda necesidad de estimación de costos en algún proyecto, por pequeño o grande que sea.

Sin embargo y en un sentido práctico, tendremos una herramienta que sirva de apoyo pero no de manera exclusiva al Ingeniero de Software o al Desarrollador de sistemas, en la elaboración de sus presupuestos o estimaciones de los proyectos de software que pretendan emprender. Que ofrezca la posibilidad de capturar independientemente cada una de las variables que proponen las métricas COCOMO II e IFPUG y que inclusive sea posible hacer, en la medida de lo coherente, combinaciones entre ellas. Que a partir del cálculo de dichas variables, obtenga sugerencias estadísticas que le ayuden a determinar el mejor camino que deberá llevar la confección de sus proyectos.

1.7 Organización de este documento.

Este documento formula una hipótesis con base en una idea, planteada y trabajada en conjunto por el autor y su asesor, misma que se sustenta y se comprueba al través de los seis capítulos que la componen: en el primero se aborda la propuesta del tema así como los antecedentes del mismo y una justificación para investigarlo.

En el segundo capítulo, se profundiza y se explican los conocimientos previos del tema. Efectuando un breve análisis de algunos de los métodos que al través de la historia de la industria de software se han propuesto para calcular las dimensiones de los desarrollos de software y que quizás actualmente se utilizan. También se aborda cómo se eligió el entorno para desarrollar tanto la investigación como la herramienta.

En el tercer capítulo se expone el diseño y la arquitectura en sí de la aplicación y los elementos que la sustentan como lo son el análisis de los requerimientos los casos de uso, los diagramas de secuencia o de estados, el diagrama de Clases, un cronograma de las actividades que se desarrollarían, un esquema planificado para la Base de Datos su diagrama Entidad Relación así como la metodología establecida y que se siguió durante el desarrollo.

En el cuarto capítulo se plantea el plan de desarrollo de la aplicación, una breve reseña de la investigación de mercado que se realizó para justificar la utilidad de la presente investigación. Un pequeño análisis comparativo de las métricas elegidas para incluir en la aplicación, así como los pormenores de la implementación del desarrollo, como lo son la configuración del entorno de trabajo, el diseño y montaje de la Base de Datos para el sistema, la correspondiente lista de cambios y una explicación de las restricciones encontradas.

En el quinto capítulo se presenta un esquema del equipo y herramientas utilizadas, un esquema de comparación con desarrollos similares existentes y la conveniencia o utilidad que tiene la herramienta propuesta ante ellos. También en este capítulo se comentan las pruebas realizadas y los resultados obtenidos.

En el capítulo seis, se comentan brevemente las metas alcanzadas, se efectúan las conclusiones a las que se llegó con la investigación y el desarrollo de la herramienta así como las visibles aportaciones de ellos para finalmente hacer un planteamiento del posible trabajo futuro en este mismo orden.

Al inicio de cada capítulo se encuentra una pequeña introducción al mismo y a excepción de los capítulos uno y seis, encontrará al final de los demás un breve resumen. Al final de los capítulos, se encuentra un listado de referencias técnicas y otro con referencias a sitios o páginas en la internet, así como la bibliografía que sirvieron de apoyo para lograr el presente trabajo.

En la parte de los apéndices se encuentran, después de un Glosario de términos y un índice de las siglas o acrónimos que se manejan a lo largo del documento. El script de la Base de Datos así como el código fuente de las principales funciones de la aplicación desarrollada, misma que se ha ofrecido presentar en una versión prototipo y finalmente un manual de usuario de dicha herramienta.

Capítulo II

Estado del Arte

2.1 Introducción

La estimación de costos del software, como ocurre con la documentación y otras tareas paralelas al desarrollo del software, es una de las actividades menos deseadas por el equipo de desarrollo de un proyecto, así como la más "pesada" de llevar a cabo. De forma parecida a lo que ocurre en la práctica de una actividad deportiva, en la que para mejorar son necesarias actividades "dolorosas" y poco gratificantes como el trabajo con pesas o el estiramiento, el desarrollo de proyectos software necesita de esas tareas "engorrosas" y como se intentará demostrar en este texto, se hacen necesarias para conseguir que el comportamiento de una organización evolucione y mejore en el tiempo de forma que su capacidad de desarrollo, así como la calidad en el trabajo realizado, etc., sean cada vez más eficientes. Actualmente, las empresas tienen en este aspecto lagunas bastante importantes, mientras los estándares proclaman la existencia y bondad de sus métodos de estimación, planificación y gestión de los proyectos, estos se reducen a veces a una planificación de los recursos en función del negocio, es decir, de lo que se supone que el cliente puede llegar a pagar por un servicio y de lo que la organización espera recibir como mínimo por el servicio prestado. Después están otros "vicios" menos importantes, pero no por ello menos perjudiciales, que redundan en la mala planificación y gestión de los proyectos de desarrollo de aplicaciones.

En este contexto de obtención rápida de beneficios como el "valor agregado", lograr contratos blindados con clientes, convertirlos inclusive en "partners", etc., provocan que la estimación de costos del desarrollo de un proyecto de software se convierta en una labor tediosa, poco agradecida y repudiada por casi todos los programadores, analistas y jefes de proyecto. Sin embargo quedará de manifiesto que con un poco de tiempo empleado en registrar las estimaciones del trabajo a realizar, el mantenimiento de esas estimaciones, su gestión y su posterior análisis de resultados se pueden obtener grandes beneficios para las empresas. Como siempre, cuando se trata de tareas similares a ésta, la clave está en lograr hacerse de virtudes tan valiosas como la paciencia, la constancia y la perseverancia.

2.2 Antecedentes

Los modelos de estimación de esfuerzo, son considerados métodos o metodologías de análisis y evaluación de los factores que intervienen en un determinado desarrollo [18]. En el caso de los elementos y agentes externos que influyen o participan en la creación y confección de aplicaciones de computadora que son susceptibles de ser cuantificados, se ha dispuesto de diversas pruebas e intentos de establecer la mejor manera de hacerlos previsibles, con la idea de poder establecer presupuestos acertados y anticipar los costos de un desarrollo [21]. Algunos de estos modelos pueden clasificarse en ciertos grupos, según su alcance tal como se aborda en el punto 2.3.

2.3 Diversos Modelos

2.3.1 Modelos algorítmicos.

Los modelos algorítmicos se apoyan en ciertos algoritmos matemáticos que realizan una estimación del coste del software en función de un número de variables consideradas como relevantes para el desarrollo de software. Se le denominan drivers de coste y de su valor depende en gran medida la calidad del producto a desarrollar y el esfuerzo y tiempo necesarios para realizarlo.

Los algoritmos más conocidos y empleados para realizar dicha estimación son:

- Modelos lineales
- Modelos exponenciales
- Modelos analíticos
- Modelos tabulares
- Modelos compuestos

Cada uno de estos modelos es tratado a continuación y al final, un resumen con las ventajas e inconvenientes de cada uno de ellos.

2.3.2 Modelos Lineales.

Tradicionalmente hemos necesitado recurrir a la formalidad de un planteamiento, así como a su consecuente demostración para que éste, sea considerado un Modelo. Así, un modelo de estimación lineal podría tener la siguiente forma:

$$E = ao + a_i x_i + \ldots + a_n x_n$$

En donde el Esfuerzo E, corresponde a la conjunción de x_i,..., x_n que son los drivers o disparadores de coste y a_o, ..., a_n que son un conjunto de coeficientes escogidos para proveer el mejor ajuste con respecto a un conjunto de datos observados en otros proyectos. De esta forma el costo de un Desarrollo, es obtenido multiplicando el esfuerzo (E) por una "constante" de trabajo.

La fórmula representa un extenso estudio por parte de la System Development Corporation (SDC) realizado a mediados de los años 60, que se realizó considerando más de 100 variables de costos utilizando datos recogidos de alrededor de 170 proyectos de software ya finalizados. Encontrando y demostrando así la utilidad del mejor modelo obtenido, mismo que tenía como base 13 variables que daba un esfuerzo medio de 40 meses—hombre y contaba con una desviación estándar de $\delta = 62\mu H$; lo cual puede ayudarnos a desestimar los modelos lineales por resultar poco factibles para una Estimación precisa y completa.

2.3.3 Modelos Exponenciales.

Un modelo exponencial se basa en una relación: $E = ao_{i^k}, \dots, a_{n^k} x_n$ Donde, al igual que antes, el conjunto x_i son los drivers del costo y el conjunto a_i son los coeficientes que provocan que la relación a los datos observados en proyectos ya concluidos se ajusten más.

Este modelo fue el utilizado por Walston Felix en su análisis (1977 en IBM), en el que las variables de costo se consideran restringidas de manera discreta (valores -1, 0 y 1) y también en el modelo Doty (1977), en el que los disparadores de los costos solo toman valores 0 o 1.

El modelo exponencial funciona bastante bien si las variables que se eligen como drivers de coste son independientes entre sí, pues de lo contrario podrían aparecer efectos colaterales y/o sobreestimación del esfuerzo (E) debido a que se estría evaluando el mismo factor varias veces, en momentos diferentes. Sin embargo la restricción de que las variables solo puedan tomar valores -1, 0 y 1 produce modelos muy rígidos e inestables.

2.3.4 Modelos Analíticos.

Este tipo de modelos toman una forma más matemática, consideran un bajo número de variables y por lo tanto son susceptibles a demostración fácilmente, sin embargo no toman en cuenta restricciones asociadas al costo de un desarrollo de software y presentan una forma como la siguiente:

$$E = f(x_i, \dots, x_n)$$

Donde $(x_i, \dots, x_n)$ son los disparadores del costo y f es una función matemática común que no necesariamente se o lineal o exponencial. Llevándonos a encontrar como ejemplos, los modelos de Halstead (1977), que tiene la forma $E = \frac{(n\#N_2\ NLog_2 n)}{2Sn_2}$ Donde n# es el número de operadores en el programa; n2 es el número de operandos; *n= n#+n₂*; N_2 es el uso total de operandos en el programa; N es el uso total de operandos y operadores; y S resulta ser una aproximación a 18. O el modelo de Putnam (1978) que encontramos con la forma $\frac{o_v 1}{3_t} \cdot \frac{4}{3} \cdot Ss = \frac{'K}{td}$, Donde *Ss* es el tamaño del producto software, *o*. es una constante, *K* es el esfuerzo de desarrollo en AH (años—hombre) y *td* es el tiempo de desarrollo en años. Este modelo evolucionará hasta el llamado modelo SLIM.

2.3.5 Modelos Tabulares.

Los modelos tabulares consisten en disponer la información de las diversas variables de costo en tablas de datos que relacionan estos valores con el esfuerzo, en las distintas fases del desarrollo de software. En ocasiones las encontramos relacionadas también con multiplicadores utilizados para ajustar el esfuerzo estimado. El modelo de Aaron Sloman (propuesto en 1969) por ejemplo, consiste en una tabla 3x3 que relaciona esfuerzo de desarrollo, duración del proyecto y dificultad del proyecto. El modelo de Wolverton (1974) estima el esfuerzo de desarrollo en base a una tabla según el tipo, dificultad y novedad del software a desarrollar.

El modelo Boeing (1977 presentado actualmente por Gary Morris, Software Engineering Manager) estima un ritmo básico de productividad como una función tabular según el tipo de software y modifica dicho ritmo de productividad mediante multiplicadores obtenidos con otras funciones tabulares que miden drivers de coste como el lenguaje empleado, el tamaño del proyecto, la novedad del mismo, su duración etc.

Estos modelos son generalmente fáciles de comprender e implementar y son también fácilmente modificables teniendo como base nuevas estrategias para el tratamiento de las variables de costos. Solo ha de tomarse en cuenta que si el número de variables es elevado, el modelo tendrá un mayor número de tablas y por lo tanto muchos más valores que calibrar, requiriendo una gran base de datos para contrastar y validar dichos valores.

2.3.6 Modelos Históricos.

Los modelos Históricos ó Experimentales, son aquellos que se basan en un conjunto de ecuaciones propuestas por agentes que cuentan con mayor experiencia en la elaboración de determinada tarea, en este caso son considerados como los expertos de la construcción del Software. Pero en realidad no tienen un sustento matemático o funcional para sus propuestas, aunque en su intento se hagan valer de diferentes estrategias.

2.3.7 Modelos Estadísticos.

Agrupa a los modelos que utilizan un análisis de regresión para establecer la relación entre el esfuerzo y los conductores de costes. Se diferencian dos tipos en función de las ecuaciones utilizadas:

a) Lineales. Los algoritmos son en esencia ecuaciones lineales.
b) No Lineales. Los algoritmos se derivan de ecuaciones complejas.

2.3.8 Modelos Teóricos.

Estos modelos se basan en teorías sobre como los seres humanos se comunican entre sí y sobre cómo funciona la mente humana durante el proceso de programación o sobre las leyes matemáticas que sigue el proceso de construcción de un producto software. Debido a su escasa formalidad y posibilidad de demostración, carecen de estudios más profundos.

2.3.9 Modelos Compuestos.

Son prácticamente una combinación de los anteriores, pueden incorporar características de los modelos lineales, exponenciales, analíticos y tabulares, para estimar el esfuerzo del desarrollo, se valen del análisis de las variables de costos, combinando dos o más enfoques. Algunos modelos de este tipo son el modelo RCA PRICES (1979) y el modelo SLIM (1979), el modelo TRW SCEP (1978), el modelo COCOMO (1981) y su evolución, COCOMO II (1997) y tal vez podría considerarse así al FFP.

Los modelos compuestos tienen la ventaja de emplear la forma funcional más apropiada para cada componente implicado en la estimación del coste del software. Sus mayores dificultades están en su complicado aprendizaje y uso y el gran número de datos que hay que contrastar y validar. Por ello, estos modelos están soportados por herramientas software que facilitan su aplicación.

2.3.10 Estimación por Analogía.

La estimación por analogía consiste en razonar por analogía o simple comparación con uno o más proyectos completados, cuál de ellos se asemeja más al proyecto que se pretende estimar. Esta técnica es similar al método de parecidos y diferencias ideado por Wolverton en 1974. Esto es, si en un plan para un nuevo desarrollo se ha detectado que podríamos estar enfrentando un problema similar a uno que anteriormente hemos superado con éxito, bien podría servir analizar el trabajo y resultados de ese proyecto anterior para la obtención de nuestros nuevos parámetros. Considerando que este método puede emplearse a nivel de proyecto, o para estimar determinados subsistemas.

La experiencia puede ser estudiada para concretar las diferencias existentes y determinar los costos asociados a esas diferencias o similitudes, según sea el caso. Una desventaja de pretender utilizar este método de estimación es que no aclara en qué medida los proyectos anteriormente desarrollados son representativos con respecto a las nuevas características que se enfrentan.

2.3.11 Estimación Top-Down

En la estimación top—Down (de arriba a abajo), contamos con un conjunto de variables para determinar el costo de un proyecto, que es extraída de las propiedades generales del producto a desarrollar. El costo total es distribuido o repartido entre los distintos componentes. Esta técnica (y también, la bottom-up) puede ser llevada a cabo en conjuración con algún otro método, dando lugar a uno compuesto.

Como la estimación se basa en la experiencia previa en proyectos completados, tendrá en cuenta cada uno de los componentes, previniendo una especial atención en la integración, generación de la documentación, administración de la configuración, etc. Lo que podría representar una de sus más grandes ventajas.

2.2.12 Estimación Bottom-Up.

Normalmente, en la estimación bottom-up (de abajo a arriba), el costo de cada componente se estima por un individuo distinto, a menudo por la persona que será responsable de su desarrollo. Después, todos esos costes son sumados para obtener el total del producto.

Este tipo de estimación es complementaria a la top-down ya que las debilidades de esta última tienden a ser las virtudes de la primera y viceversa. Ya que la estimación bottom-up se centra en cada componente, pierde de vista costes globales como son la integración, gestión de configuración, gestión de calidad, etcétera; asociados al desarrollo de un proyecto. Por lo tanto, este tipo de estimaciones suelen ser un poco optimistas.

2.4 Estado del Arte.

Al través de las últimas décadas, se ha avanzado vertiginosamente en el desarrollo científico y tecnológico en todas las áreas y hemos sido testigos de que las computadoras han jugado un papel significativo. Por lo tanto sería de esperarse que los desarrollos de software también hubiesen alcanzado tal grado de sofisticación, pero en el caso de algunos países como el nuestro, nos hemos estado conformando con ser solo usuarios más que participantes del avance tecnológico en computación.

Lamentablemente, el tema se convierte para la industria de software en un recurso teórico del que se pudiera sacar partido en algún momento, de cara a convencer al cliente de los retrasos en la entrega de un producto y como ocurre en la mayoría de estos casos, solo las universidades y ciertos centros privados exponen sus ideas e investigaciones respecto a este tema de forma objetiva, útil y provechosa.

Así, cabe destacar la labor de los profesores de las materias relacionadas con la Ingeniería de Software, quienes a pesar de las amplias críticas del resto de sus colegas, le dan a este tema la importancia que requiere. Personalidades como Somerville [1], Florac [2], o Pressman [4], quienes han dedicado su vida a la aplicación científica de la ingeniería en la creación de software; o como Boehm [3], Albecht [1]y Horowitz [7], quienes además han aportado significativamente al enfoque presupuestal y estimativo de los desarrollos de software participan activamente en la investigación científica en este rubro.

Existen también numerosas organizaciones privadas involucradas en la investigación de esta actividad entre las que se encuentran reconocidas organizaciones como la IFPUG (International Function Points Users Group)[26], QSM (Quantitative Software Management, Inc.) [27], TotalMetrics [29][30], AEMES (Asociación Española para la Medición y Estimación de Software)[31][32], Galorath[44][45], SoftwareMetrics [37], entre muchas otras, que además ponen en práctica sus propias investigaciones así como las de universidades como Princeton, la USC (University of Southern California)[38][39], Carnegie-Mellon y algunas más.

A continuación, se describen de forma genérica los principales métodos y procedimientos de estimación de proyectos software, Tabla 1, agrupados según su esquema fundamental de operación. Primero con base en su tipología, posteriormente se enumeran unos cuantos modelos y finalmente se hace mayor hincapié en el modelo que sirve de base para este proyecto fin de carrera.

EXPOSITORES	PROCEDENCIA	METODOLOGÍA	DESARROLLO	FECHA
Sandro Fonseca R. & Jennifer H. Torres	http://ingenieriasoftware1.blogspot.es/i2009-06/	Function Points & COCOMO	desarrollada en Visual Basic	2009 [46]
Fernando Gómez Fernández	Universidad Politécnica de Madrid.	Con base en modelo SLIM.	desarrollada en Visual Studio .NET (C#)	2008
Janet Russac & Betsy Clark	Management Reporting Committee,	IFPUG	Procedure for Obtaining Certification	2004 [18]
Costar™ 7.0	Softstar Systems.com	COCOMO II	desarrollada en Builder C++	2004 [36]

Tabla 1. Estado del Arte: Herramientas de software que aplican métricas de estimación de costos

Primeramente podemos observar que no existe una herramienta de software basada en IFPUG [17] como tal, en segundo lugar encontramos que no hay tampoco una herramienta que intente fusionar las metodologías buscando una mejora a las mismas y podemos decir por último que los intentos por automatizar los procedimientos de Cocomo II [13], aunque no son pocos, han ofrecido solamente una mediana ayuda a los expertos en Ingeniería de Software, presentando aplicaciones de software sumamente difíciles de interpretar y por lo tanto, de utilizar. Como lo muestra la figura 24, los entornos gráficos que ofrecen son muy simples y con escaza ayuda para poder utilizarlo.

2.5 Modelo COCOMO

El modelo COCOMO (de COnstructive COst MOdel), es actualmente el modelo más empleado y uno de los más consistentes, fue desarrollado por Barry Boehm a comienzos de los años 80, apoyado en una base de datos de 63 proyectos de software. Este modelo reúne virtudes de algunos de los modelos anteriores al mismo tiempo que intenta corregir ciertos errores de concepto en los mismos. El primer paso de este método es la obtención de las fórmulas para el cálculo del esfuerzo de desarrollo del proyecto (en meses-hombre, MM) y la duración del desarrollo (en meses, TDEV). Estas fórmulas se

obtienen según dos factores: El modelo de desarrollo, que puede ser orgánico, semilibre o rígido y el modelo de estimación, que puede ser básico, intermedio o detallado. Las características de los distintos modos de desarrollo son las siguientes:

- Modo orgánico
  - Pequeños equipos de desarrollo
  - Entorno estable y familiar
  - Personal con amplia experiencia
  - Ningún tipo de restricciones
  - Aplicaciones de pequeño tamaño (50 KDSI)
  - Facilidad del usuario a acomodarse a posibles diferencias entre especificaciones y producto.
- Modo Semi libre
  - En general, una mezcla de características de los modos orgánico y rígido
  - Personal con un nivel de experiencia intermedio en proyectos similares
  - Proyectos con interfaces muy rigurosas y otras muy flexibles
  - Proyectos que se desarrollan dentro de unas limitaciones muy estrictas
  - El producto se explota dentro de un entorno muy restringido tanto de hardware, como software y así mismo de procedimientos operativos muy estrictos.
  - Mezcla de gente experta e inexperta
  - Tamaño máximo 300 KDSI Modo rígido
  - Alto coste de modificaciones
  - No se permiten cambios
  - Proyectos en áreas poco conocidas
  - Sin limitación de tamaño

En cuanto a las características de los distintos modelos de estimación, se distinguen:

- Modelo básico
  - Productos de tamaño pequeño/ medio
  - Personal de la propia empresa
  - Generalmente en modo orgánico
- Modelo intermedio
  - Para estimaciones más exactas
  - Empleo de 15 atributos drivers de coste con influencia en la estimación
- Modelo detallado
  - Para grandes proyectos software
  - Multiplicadores de esfuerzo por fases
  - Jerarquía en al menos tres niveles: sistema, subsistema y módulo.

La fórmula para hallar MM está en función del tamaño estimado del proyecto en instrucciones de código fuente y TDEV se halla en función de MM. Para el modelo básico de estimación con esto ya sería suficiente, si se requiere una estimación de modelo intermedio o detallado habrán que considerarse 15

variables conocidas como *"drivers de coste"* que refinarán los datos obtenidos mediante las ecuaciones. Estos 15 multiplicadores se agrupan de la siguiente forma:

- Atributos del producto
  - Fiabilidad requerida del software
  - Tamaño de la base de datos
  - Complejidad del producto
- Atributos del ordenador
  - Limitaciones en el tiempo de ejecución
  - Limitaciones de memoria principal
  - Volatilidad de la máquina virtual
  - Tiempo de respuesta
- Atributos de personal
  - Capacitación de los analistas
  - Experiencia en aplicaciones
  - Capacitación de los programadores
  - Experiencia en la máquina virtual
  - Experiencia en el lenguaje de programación
- Atributos del proyecto
  - Prácticas modernas de programación
  - Uso de herramientas
  - Limitaciones en la planificación

Una vez que se estima el valor de cada uno de estos parámetros, se multiplican entre sí y el factor total obtenido se multiplica por el esfuerzo MM calculado mediante la fórmula nominal correspondiente. Con el MM corregido se halla el TDEV según las mismas fórmulas que para el modelo básico.

El modelo COCOMO permite también, mediante unas tablas, ajustar el esfuerzo a las distintas fases de desarrollo según el modelo de ciclo de vida en cascada. Y conociendo la fracción de instrucciones del código fuente del producto que cambian en un año por adiciones o modificaciones, se tendrá el esfuerzo de mantenimiento del producto.

2.6 Modelo SLIM

La resolución del problema se basa en el modelo SLIM (Software LIfecycle Management) presentado por Lawrence H. Putnam [SLIM, 1991], propone que una vez que se ha establecido el conjunto de variables que han de considerarse para calcular, solo será cuestión de aplicar correctamente las fórmulas que él mismo plantea. En este modelo es muy frecuente encontrar el término *"La Ecuación del Software"*

Que se refiere a que la cantidad de trabajo que se puede encontrar en cualquier producto de software, puede verse como el resultado del esfuerzo realizado en un periodo de tiempo y se puede escribir como

$$Producto = (Constante) \cdot Esfuerzo \cdot Tiempo$$

Donde Producto, representa cierta medida sobre la funcionalidad del mismo y se cree proporcional al producto (Esfuerzo · Tiempo). La medida SLOC suele ser una medida habitual de la funcionalidad.

Esfuerzo, representa el trabajo humano, medido en personas—hora, personas—mes ó personas—año. Y Tiempo, representa la duración del trabajo, medido en meses o años.

Constante, es un factor de proporcionalidad. Una vez establecidas las otras tres variables, esta constante permite igualarlos. Sin embargo parece que la cantidad de producto depende también de "cómo se hacen las cosas", puesto que con el mismo esfuerzo y tiempo y dependiendo del entorno de trabajo, podremos conseguir mayor o menor cantidad de producto.

La ecuación anterior tiene mayor sentido si la expresamos como:

$$Producto = Productividad \cdot Esfuerzo \cdot Tiempo$$

En donde podemos observar que el valor de Productividad, podría deducirse si conocemos los otros términos, como:

$$Productividad\ \frac{Producto}{(Esfuerzo\ \bullet\ Tiempo)} \qquad (1)$$

Aunque de esta manera la Productividad no está del todo definida, suponemos que incluye un conjunto de factores que afectan a toda la organización, como lo son:

- la gestión del proyecto.
- la utilización de buenos requerimientos, diseños, codificaciones, inspecciones y pruebas.
- el nivel del lenguaje de programación empleado.
- el estado de la tecnología al momento del desarrollo.
- la experiencia de los miembros del grupo de trabajo
- la complejidad de la aplicación a desarrollar.

En un determinado momento del proyecto, el valor de la Productividad es fijo. Sin embargo, la evolución en el tiempo y la acción de la dirección puede hacer cambiar su valor. Esta productividad es un conjunto de factores y se puede denominar "parámetro de productividad del proceso".

2.7 Modelo COCOMO II.

Nace como la adaptación del exitoso modelo COCOMO a los nuevos sistemas, herramientas y técnicas de desarrollo. Utilizando la base de su predecesor, COCOMO II implementa nuevos conceptos para ajustar más la estimación a las características de los proyectos, esto es por lo que se definen tres sub modelos de estimación:

- Composición de aplicaciones: Utilizado para estimaciones en las primeras etapas del ciclo de vida de un proyecto software.
- Diseño anticipado: Preparado para sistemas en diseño, produce estimaciones tan fiables como el grado de evolución se haya alcanzado en el diseño de la aplicación.
- Post-Arquitectura: Empleado para sistemas completamente diseñados y como paso previo al comienzo de la etapa de desarrollo, así como para el seguimiento del esfuerzo invertido en el desarrollo del producto.

Respecto del anterior modelo, también se han incluido nuevos *"drivers de coste"* y eliminado algunos otros de los que empleaba. También y debido a la capacidad de proceso de las nuevas computadoras, las herramientas de calibración permiten obtener resultados más ajustados y manejar bases de datos de proyectos más grandes. De esta manera las fórmulas que propone el Modelo COCOMO II son mucho más confiables. Además, la clasificación del modo de desarrollo en orgánico, semilibre, o rígido, se realiza ahora de forma más compleja con base en algunos factores de escala:

- Precedencia: experiencia en aplicaciones del mismo tipo.
- Flexibilidad de desarrollo: grado de sujeción del desarrollo a tiempos y requisitos.
- Resolución de arquitectura y riesgos: identificación de riesgos en la aplicación.
- Cohesión del equipo: nivel de integración del equipo de desarrollo.
- Madurez del proceso: experiencia en el modelo de desarrollo.

De acuerdo con el primer modelo de COCOMO, esta evolución maneja también los rangos de constantes que se utilizarán para las evaluaciones con el resto de las variables según el tipo de desarrollo que se está pronosticando, considerando los valores de la Tabla 1 para el modelo básico.

Tabla 2. Coeficientes para los modos de desarrollo según COCOMO II

MODO	a	b	c	d
Orgánico	3.20	1.05	2.50	0.38
Semilibre	3.00	1.12	2.50	0.35
Rígido	2.80	1.20	2.50	0.32

Del mismo modo la clasificación de los manejadores de costo mara el modelo intermedio como en el predecesor de esta métrica que se pueden agrupar en al menos cuatro categorías, podemos ponderarlas según se muestra en la Tabla 2, considerando siempre el grado de subjetividad que puede existir en este tipo de parámetros.

En adición a los que proponía la primera versión del modelo COCOMO, en la revisión del 2000 Bohem presenta dos más, siendo a partir de entonces 17 modificadores.

Tabla 3. Multiplicadores de Esfuerzo

Relacionados con el producto:		**Valor ME**	**COQUALMO**
RELY:	Fiabilidad exigida al software,	1.10	2.10
DATA:	Tamaño de la base de datos	1.28	1.25
CPLX:	Complejidad del producto;	1.17	2.00
RUSE:	Desarrollo para ser reutilizado;	1.15	1.05
DOCU:	Cantidad de artefactos que deben ser documentados;	1	1.40
Relacionados con la plataforma de desarrollo:			**COQUALMO**
TIME:	Exigencias sobre capacidad de ejecución;	1.11	1.25
STOR:	Exigencias sobre almacenamiento del sistema;	1.05	1.20
PVOL:	Volatilidad de la plataforma;	0.87	1.50
Relacionados con personal			**COQUALMO**
ACAP:	Capacidad de los analistas;	0.71	-
PCAP:	Capacidad de los programadores;	0.76	1.75
PCON:	Volatilidad del personal;	0.81	1.70
APEX:	Experiencia previa en área de aplicación;	0.81	1.30
PLEX:	Experiencia previa con la plataforma;	0.85	1.35
LTEX:	Experiencia previa con el lenguaje y herramientas de desarrollo;	0.84	1.50
Relacionados con el proyecto			**COQUALMO**
TOOL:	Uso de herramientas de software;	0.78	1.58
SITE:	Desarrollo en localidades distribuidas;	0.86	1.40
SCED:	Exigencias sobre el calendario.	1.00	1.40

En la Tabla 4 podemos identificar los factores que tienen que ver más con el capital humano y su grado de integración con la organización que está a cargo del Desarrollo, conocidos como Factores de Escala:

Tabla 4. Factores de Escala para COCOMO II.

Factor de Escala		**Descripción**	**COQUALMO**
(PREC)	Precedencia	Refleja la experiencia previa de la Organización	1.55
(FLEX)	Flexibilidad de desarrollo	Grado de Flexibilidad en los procesos del Desarrollo	-
(RESL)	Resolución de riesgos	Manejo de Riesgo y Arquitectura	2.00
(TEAM)	Cohesión de Equipo	Cohesión del Equipo de Desarrollo	1.40
(PMAT)	Madurez del Proceso	Refleja la Madurez de los procesos en la organización.	2.50

Según los propulsores de este modelo pretenden presentarlo como único y mejor al resto, dado que al evaluar el tamaño, se introduce el concepto de Punto Objeto, para manejar la complejidad de los bloques de código, basándose en herramientas y lenguajes de 3a generación, asumiendo que debido a ello, los Puntos Función podrían considerarse obsoletos.

2.8 Modelo COCUALMO.

Una estimación del número esperado de defectos La familia de modelos de COCOMO II incluye COQUALMO (COcomo QUAlity MOdels) conformado por dos modelos, un modelo de inyección de defectos y un modelo de remoción de defectos. La salida del modelo de inyección de defectos es el número de defectos no triviales, lo que incluye:

Defectos críticos: causan caída del sistema o pérdida irrecuperable de datos o pone en riesgo a la gente;
Defectos graves y moderados: impiden que ejecuten correcta o apropiadamente funciones críticas del sistema.
Estos modelos están bajo desarrollo. Según una primera aproximación, las tasas de inyección base de defectos no triviales son:

- 10 defectos de requerimientos por KSLOC;
- 20 defectos de diseño por KSLOC;
- 30 defectos de codificación por KSLOC.

Nótese que COQUALMO considera que se genera una tasa de defectos por tamaño de código, a diferencia de TSP que considera que se genera por tiempo de desarrollo. Las tasas base de COQUALMO, pueden ajustarse con casi todos los multiplicadores y factores de ajuste salvo FLEX. Nótese también que para un desarrollo estimado de 3 KSLOC correspondiente al desarrollo de Delta Pensum entre Septiembre 2000 y Marzo 2001, este modelo arroja un estimado de 30 defectos de requerimientos, 60 de diseño y 90 de codificación. Desafortunadamente, la falta de una definición clara para lo que constituye un defecto de requerimiento o de diseño, nos deja con un estimado de entre 90 y 180 defectos por KSLOC (90 corresponde a solo tomar los defectos de codificación, 180 a tomar los tipos de defectos; el modelo COQUALMO no explica claramente cuál de las opciones es la correcta). Utilizando la tasa preliminar dada en clase de 5 defectos por falla, esto arroja un número aproximado de entre 18 y 36 fallas, que correlaciona bastante bien con la cifra de 24 fallas, calculadas a partir de la adaptación de TSP. Sin embargo, las cifras lucen altas si se considera que corresponde al rango base, sin ajuste por multiplicadores.

No entraremos en detalle en el modelo de remoción de defectos. Basta con indicar que la tasa de remoción de defectos se ve afectado por inspecciones, herramientas de análisis y pruebas (de la ejecución). COQUALMO estima que la tasa residual de defectos puede variar entre 1.57 y 60 defectos por KSLOC, Las pruebas contribuyen a remover entre 23 y 88% de los defectos (pero, solo de codificación).

2.9 Modelo Puntos Función.

El tamaño de un software podría medirse en términos de los bytes o espacio que ocupa en disco, o tal vez como el número de programas ó quizás por el número de líneas de código, o por la funcionalidad que proporciona, o simplemente por el número de pantallas o reportes que entrega.

Sin embargo, al querer medir la relación con el esfuerzo que implica realizarlo debemos considerar que al final de cuentas una aplicación de software es un conjunto de líneas de código que se ejecutan en una computadora y sin embargo mucho del costo de producir ese software no está directamente relacionado con la codificación ya que se ha demostrado que esto representa apenas entre el 20 y 25% del costo total de un desarrollo de software. Elementos como la administración del proyecto, el nivel de detalle de la planeación, el diseño, la documentación técnica o de pruebas y las pruebas en sí mismas también deben considerarse.

Por otro lado, hoy existen una gran variedad de lenguajes y herramientas para producir software, lo que ha provocado que pueda generarse la misma funcionalidad de un determinado software, utilizando distintos lenguajes de programación dando como resultado un número de líneas de código distintos y por lo tanto, con un esfuerzo distinto.

Las implicaciones de ello podemos observarlas mediante un sencillo ejemplo: Supongamos que se requiere cierta funcionalidad en una aplicación de software y se presenta la opción de solicitarlo a dos empresas. La empresa A utiliza un lenguaje que requeriría 7000 líneas de código. La empresa B utiliza otro lenguaje que requeriría aproximadamente 4000 líneas de código. A simple vista parecería que con la empresa A tendría “más” trabajo, pero con la empresa B podría salir más barato, dado que tienen que hacer menos líneas de código. La comparación se complica si los programadores de ambas empresas producen líneas de código a distintas velocidades, si acaso los de la compañía A producen casi el doble de código por día que los de la compañía B, tendríamos como resultado que la línea de código de la empresa A podría costar mucho menos que la de la compañía B, Por lo tanto el costo total que ofreciera la compañía B, sería menor o el programa resultante podría tener menos posibilidad de errores, etc.

Un análisis tan superficial y subjetivo de un problema así, podría resultar además de engañoso, totalmente erróneo, si tomamos en cuenta que lo que al final requiere el cliente, no es saber cuánto se tardó, o qué tan complejo resultó ser, sino que se han visto satisfechos los requerimientos iniciales del usuario final.

Ante ello se propuso una métrica que se basara en la funcionalidad, más que en otros detalles y que tiene como puntos relevantes:

- El grado de Independencia Tecnológica.
- Su Simplicidad
- Enfoque en la Utilidad
- Grado de cumplimiento de los objetivos.
- Su Consistencia

Una vez que contamos con información precisa del tamaño de un software, será posible utilizar ese dato para distintos propósitos. Entre ellos dar seguimiento a la Calidad de dicho software, observando:

- La Administración de la productividad.
- La Administración de la calidad.
- La Comparabilidad con proyectos semejantes.

El método de puntos función propuesto originalmente por Alan Albrecht [10], se basa en los dos componentes intrínsecamente relacionados en todo sistema de información, por un lado el volumen de información que maneja el sistema y por el otro el grado de dificultad técnica que podría representar al usuario final, dicha información. Ante ello el método propone clasificar estas relaciones en cinco diferentes tipos o grados de complejidad entre ellas. Están por un lado las externas lógicas, que son las Entradas, Salidas y de Búsqueda; por otro lado las Internas lógicas, constituidas por los Archivos del sistema y por último las Interfaces hacia y desde otros sistemas.

A tales relaciones son a lo que Albrecht llama funciones o "puntos función". Estos componentes proporcionan un cierto puntaje para la métrica y dependiendo de su complejidad se clasifican en "simple", "promedio" y "complejo". Una vez evaluados dichos elementos, son frecuentemente expresados como "Puntos de Función Ajustables" UFP (o sin ajustar, por su nombre en inglés Unadjusted Function Points). Para lograr una medición de tales aspectos, Albrecht propone una tabulación similar a la que podemos observar en la Tabla 4.

Tabla 5. Conteo de Puntos Función Ajustables.

Descripción	Función: Nivel de Información de Procesamiento			Total
	Simple	Promedio	Compleja	
Entrada	__ * 3 = __	__ * 4 = __	__ * 6 = __	____
Salida	__ * 4 = __	__ * 5 = __	__ * 7 = __	____
Archivo int.	__ * 7 = __	__ * 10 = __	__ * 15 = __	____
A. Interfase	__ * 5 = __	__ * 7 = __	__ * 10 = __	____
Búsq. Comp.	__ * 3 = __	__ * 4 = __	__ * 6 = __	____
Total de Puntos Función Ajustables:				

Si se opta por la técnica de evaluar el factor de complejidad del desarrollo, también es posible efectuar una tabulación, según muestra la Tabla 5, para determinarlo a partir de la participación o combinación de diferentes grados de intervención de 14 componentes, graduados también en grados desde "no presente" o "sin influencia" (con valor cero), hasta "fuertemente" o "totalmente" (con valor de cinco).

Tabla 6. Factor de Complejidad Técnica.

ID	Característica	GI	ID	Característica	GI	Valores para el GI
C1	Comunicación de Datos		C8	Actualización en línea		No Presente o Sin Influencia = 0 Influencia Insignificante = 1 Influencia Moderada = 2 Influencia Media = 3 Influencia Significativa = 4 Influencia Fuerte o total = 5
C2	Funciones Distribuidas		C9	Complejidad del Proceso		
C3	Performance		C10	Re Usabilidad		
C4	Arreglo más utilizado		C11	Facilidad de Instalación		
C5	Taza de transacciones		C12	Facilidad de Operación		
C6	Entrada de datos en línea		C13	Multi sitio		
C7	Eficiencia de Usuario Final		C14	Facilidad de Cambios		
				Grado Total de Influencia:		

En esta tabla 5 podemos observar de qué manera es posible calcular el Factor de Complejidad Técnica con la fórmula en (2)

$$FCT = 0.65 + (0.01 \cdot GTI) \qquad (2)$$

Y obtener el total de UPF (Unadjusted Function Points) de manera similar con la fórmula en (3)

$$FP = CT * \left(0.65 + \left(0.01 \cdot \sum_{i}^{n} (fi)\right)\right) \qquad (3)$$

Donde:

CT (cuenta total) es la suma de todas las entradas obtenidas de la tabla 4.
fi donde i puede ser de uno hasta los 14 valores de ajuste de complejidad basados en las respuestas a las siguientes preguntas:

1. ¿Requiere el sistema copias de seguridad y de recuperación fiables?
2. ¿Se requiere comunicación de datos?
3. ¿Existen funciones de procesamiento distribuido?
4. ¿Es crítico el rendimiento?
5. ¿Se ejecutaría el sistema en un entorno operativo existente y fuertemente utilizado?
6. ¿Requiere el sistema entrada de datos interactiva?
7. ¿Requiere la entrada de datos interactiva que las transacciones de entrada se lleven a cabo sobre múltiples pantallas u operaciones?
8. ¿Se actualizan los archivos maestros de forma interactiva?
9. ¿Son complejas las entradas, las salidas, los archivos o las peticiones?
10. ¿Es complejo el procesamiento interno?
11. ¿Se ha diseñado el código para ser reutilizable?
12. ¿Están incluidas en el diseño la conversi6n y la instalaci6n?
13. ¿Se ha diseñado el sistema para soportar múltiples instalaciones en diferentes organizaciones?
14. ¿Se ha diseñado la aplicaci6n para facilitar los cambios y para ser fácilmente utilizada por el usuario?

La métrica de Puntos Función, se define como su nombre lo dice, por una métrica funcional dado que se enfoca a evaluar el grado en que el software desarrollado proporciona funcionalidad al usuario. Considerada por algunos como una métrica para establecer el tamaño y complejidad de los sistemas informáticos que se basa en la cantidad de funcionalidad requerida y entregada a los usuarios, o a veces entendida también como la única métrica que ve a los Puntos Función como aquellos que miden el tamaño lógico o funcional de los proyectos o aplicaciones de software con base en los requerimientos funcionales solicitados por un usuario.

2.10 Elección del entorno de Desarrollo.

Debido a que se pretende automatizar al menos una metodología considerada compuesta, lo que conllevará dos o más de las anteriormente comentadas, se entiende que al final podría resultar un sistema de información, con todos sus elementos. Aunque tal vez no sea un sistema muy grande o sofisticado, será conveniente utilizar un IDE que ofrezca cierto nivel de facilidad de acceso a herramientas gráficas que provean apoyo en cuanto al diseño de una interfaz amigable.

Considerando también que es un desarrollo individual y de índole académico, tampoco está contemplado hacer alguna inversión importante tanto en software como en hardware, por lo que será preferible utilizar alguna plataforma de programación de acceso libre y que sea de uso más bien común para que en determinado momento esté al alcance de cualquier interesado en el tema.

En la actualidad existen innumerables herramientas de software de desarrollo y de programación de bases de datos, por los que no hay que pagar una licencia y en los que se puede confiar cabalmente para realizar el trabajo que se requiere para esta investigación.

2.11 Resumen.

A través de los modelos encontrados y de las aportaciones que en esta materia han hecho todas las personas, las que han sido mencionadas y aun las que no figuran en las investigaciones, pero que han sido de igual forma partícipes del logro de todas ellas; pudimos observar que algunas de los llamados métodos compuestos, conforman precisamente las métricas más completas y también las más elaboradas.

En el presente capítulo se hizo un trabajo de búsqueda, recopilación y comparación entre dichos métodos y se plantea llevar a cabo un análisis de algunos de ellos para identificar cuáles son los más precisos. Entonces poder proponer una mejora será el trabajo que se abordará en los capítulos siguientes y dicha propuesta conformará la prioridad de la presente investigación.

Capítulo III

Análisis y Diseño de la Aplicación.

3.1 Introducción.

Actualmente, existen pocas situaciones del mundo real que no sean susceptibles de ser automatizadas o simuladas y que por lo consiguiente pueden ser computarizadas y manejarse de manera interactiva por cualquier usuario. Para que esto último sea posible, debe garantizarse en cierta forma, que todo aquello que decida automatizarse y llevar a una máquina computadora sea también comprensible y hasta cierto punto manipulable además de entender que puede aprenderse a través de dichas aplicaciones.

Intentar crear un modelo de parametrización[1] de aplicaciones de software casi siempre se traduce en pretender encontrar las variables estratégicas del proceso de su desarrollo y derivar del comportamiento de dichas aplicaciones, un sistema de variables y ecuaciones e introducirlas en una computadora para su estudio y análisis.

Como ya se ha establecido, encontrar dichas variables o factores ha sido el objetivo de innumerables estudios acerca de la estimación de proyectos de software. En la mayoría de los casos estos factores han sido ordenados según su importancia, pero no medidos de forma precisa. La ordenación es siempre subjetiva, depende del individuo que la realiza o de casos particulares.

Al proponer una herramienta de software que ayude a calcular y evaluar mediante uno o varios modelos de medición, cuanto ha de tardarse o cuánto podría costar o finalmente lo que ha de representar en esfuerzo la creación de un nuevo proyecto de software; lo que se pretende es medir y calcular dichos factores de forma objetiva y de la manera más precisa posible, para que ante la mirada de cualquier observador sea posible determinar el grado de influencia y certidumbre que tendrá la modificación de cualquiera de tales variables, en el desarrollo entero.

[1] El término parametrización no existe en la RALE, pero se utiliza en éste documento para establecer una manera de enumerar rangos que se utilizarán como valores o parámetros en la asignación de variables dentro de la herramienta de software.

3.2 Arquitectura de la Aplicación.

En la práctica, el desarrollo de proyectos pocas veces se planifica debidamente y menos veces aún se documenta. Inclusive, muchos diseños deliberadamente omiten una documentación debido a que su aparición y aplicación es de corta duración y una vez que un plan o especificación está fuera de tiempo, es inútil, por ejemplo, intentar aplicar cambios en el código fuente o en el diseño ya que rápidamente caen en la obsolescencia. Decidir lo que es más importante de un proyecto es una habilidad esencial de un arquitecto competente.

Para el presente desarrollo tomaremos como base una Arquitectura centrada en el proceso, misma que en teoría se lleva a través de ciertos pasos definidos para lograr abarcar el ciclo de vida completo del sistema. Entendemos así que este proceso se basa en estándares claves de la Ingeniería de Software.

- Prevención del Sistema.- Establecer una visión o panorama del sistema completo
- Análisis de los requerimientos.- Se busca satisfacer Necesidades.
- Planificación y Captura de Requisitos.- Se plantea el modelo a seguir (casos de Uso)
- Garantizar la simplicidad y la claridad.- Con base en el Modelo aprobado, se determina la mejor ruta, en aras de que el resultado sea un sistema fácil y amigable.

3.3 Recopilación y Análisis de los Requerimientos.

Se propone desarrollar una herramienta (al menos en una versión preliminar ó “prototipo”) que sea capaz de combinar y aplicar a elección del usuario, los dos principales métodos de medición que existen en el campo de la ingeniería de software, que son COCOMO II e IFPUG, para auxiliarlo en la planeación de un nuevo proyecto de Desarrollo de Software.

Según la herramienta propuesta, se debe permitir al usuario establecer el ámbito del proyecto, establecer objetivos y metas del proyecto, así mismo debe sugerir o permitir que el usuario ingrese estrategias, políticas y procedimientos para alcanzarlos. Clasificando tenemos:

1. Establecer un ambiente amigable y entendible.
2. Ofrece elegir entre las dos metodologías propuestas.
3. Debe delimitar las tareas o actividades del software.
4. Permitir Ingresar los recursos a estimar (hardware, software, personal, etc.…)

5. Permitir elegir el valor para las diferentes variables, en rangos preestablecidos.
6. Proporcionar al usuario una predicción basada en un modelo probabilístico (no determinístico)
7. Pueda el usuario proponer distintos valores considerando que el objetivo de la estimación, es reducir los costos e incrementar los niveles en aspectos de servicio y de calidad.

Como resultado de estas opciones deberá obtenerse información que permita establecer:

1. El esfuerzo humano requerido para completar el proyecto.
2. El tiempo que habrá de tomar (incluso cambiando la cantidad de personal).
3. El Costo total del proyecto (desglosando rubros importantes)

Debemos considerar que el que exista determinado software, no significa que haya sido resuelto un problema en su totalidad, pues estaríamos vanamente suponiendo que por el hecho de que se cuente con una determinada herramienta de software, estaremos agotando la problemática pues ésta no puede ser tan general como para abarcar todos los aspectos de la Ingeniería de Software, pues basta como ejemplo, observar cómo para cada tarea de las más cotidianas actividades de una oficina, existen más de una versión y que son ofertadas por más de un fabricante de aplicaciones de computadora que de alguna forma coadyuvan en la realización de todas esas actividades, pero que están muy lejos de resolver el trabajo del oficinista y más lejos aún de suplantar al elemento humano. Incluso si el usuario no está medianamente capacitado en el manejo de dichas aplicaciones, lejos de representar una ayuda o facilidad para desarrollar con mayor eficiencia su trabajo, se vuelve un verdadero obstáculo que en lugar de permitirle avanzar en su trabajo, puede llegar a perderlo por no hacer ni siquiera, lo que hacía antes de contar con la nueva herramienta.

Así, una herramienta de software entre más general se pretenda, deberá abarcar muchas más funciones, sería más grande y difícil de aprender a manejar en su totalidad, por lo tanto, en el presente proyecto se plantea realizar una herramienta de software que sirva como auxiliar en la etapa planeación de un proyecto de software. Esta herramienta no será un asistente tipo CASE, que resuelva el trabajo del ingeniero de sistemas, tampoco sustituirá alguna de sus tareas.

Al contrario, tal como sucede con el oficinista que previamente deberá haber aprendido a realizar cada una de sus actividades de la manera tradicional y además conocer al menos de manera superficial el manejo de una computadora, para después poder realizar dichas tareas con ayuda de una herramienta de software; el Ingeniero de Sistemas, podrá valerse de dicha herramienta solo si sabe cómo desarrollar sus actividades aún sin valerse de ninguna.

No se pretende que el usuario de la herramienta propuesta, sea específicamente un experto desarrollador o un reconocido ingeniero de software, para que pueda utilizarla, pero será verdaderamente inútil intentarlo, si acaso no se tiene una noción no tan mínima de los aspectos que se manejan en la administración de sistemas y en el diseño y planeación de un proyecto de software. Servirá de mucho si no solo ha realizado ya algunos proyectos de software, sino que además esté familiarizado con el conjunto de parámetros susceptibles de medición en dichos proyectos y que tenga algún conocimiento de las métricas más utilizadas hoy en día para la estimación de costos de proyectos de esta índole así como de algunas de las variables que estas métricas manejan.

Por todo lo anteriormente expuesto, en los siguientes apartados se dará detalle de los elementos que contendrá dicha herramienta, asumiendo que lo que no se encuentre en el glosario de términos, es porque el lector ya lo conoce o al menos sabe que tales elementos se manejan en la ingeniería de software.

3.4 Modelado.

Es posible afirmar que Modelar, se ha convertido en la actividad central de todo ser humano, puesto que a lo largo de su evolución, ha alcanzado niveles asombrosos del conocimiento. Si se analiza desde el punto de vista que saber es organizar y que por ende organizar es modelar, podría considerarse que cada quien construye su propia comprensión del mundo que le rodea y con base en su propia experiencia. Cada cual continúa creando y cambiando esa comprensión al través de su vida. A dicha comprensión que se le llama “modelo” del mundo.

Tanto la comprensión informal como la comprensión formal de la ciencia son modelos, algunos con base en teorías, construidos mediante el proceso de representación y abstracción de la realidad.

3.4.1 Caso de Uso General

En el presente trabajo, tenemos como primer requerimiento el de ofrecer dentro de una herramienta, un ambiente entendible y amigable, por lo que después de una breve presentación, el software ofrecerá como primera vista una explicación de las diferentes variables que evalúa cada métrica COCOMOII e IFPUG y la opción de seleccionar alguna de ellas.

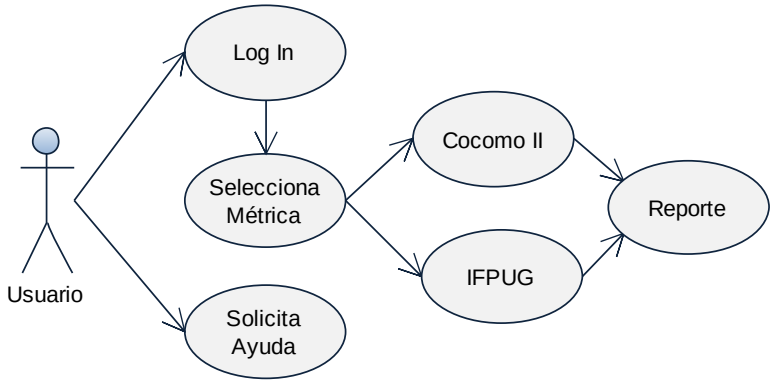


Figura 1. Representación UML de la primera vista de la Aplicación.

En la figura 1 se muestra una abstracción de los elementos que componen en resumen, a la aplicación, en cuanto el usuario interactúa con ésta. Éstos elementos son el inicio del sistema o la aplicación, momento desde el cual es posible finalizarlo o seleccionar una métrica para efectuar un llenado de ciertos datos para obtener una aproximación o una estimación. Es posible proporcionar éstos datos en un formato especial para el caso de uso, como el que se muestra en la Tabla 5, al que se conoce como Escenario del Caso de Uso.

Tabla 7. Escenario de Caso de Uso General de la herramienta.

Identificadores:	
Selección de métrica	
Actores:	**Flujo de eventos:**
Usuario del sistema	Elige una métrica
	Selecciona valores para las variables
	Acciona control de ejecución
Condiciones Iniciales:	**Condiciones de Paro:**
Variables a cero	A elección del Usuario
Condiciones de Salida:	**Requerimientos especiales:**
A elección del usuario	Ninguno

3.4.2 Diagrama de Estados o de Secuencia.

Elegir una de esas “opciones” se representa en la figura 2, que es un diagrama de los estados o “momentos”, en los que podría estar la aplicación, en caso de elegir “una Métrica” en lugar de “finalizarla”.

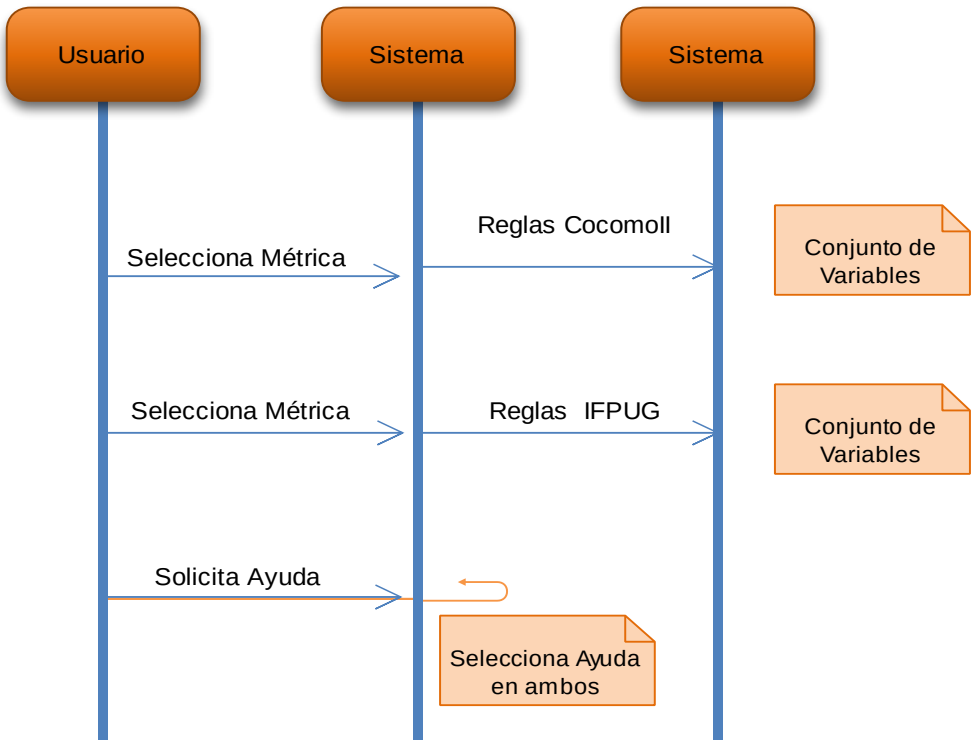


Figura 2. Diagrama de estados de las fases de la Aplicación.

Puede observarse que el sistema planteado es tan sencillo como lo representa el diagrama de caso de uso (Figura 1). Se tratará de una interfaz o ventana inicial, que permite elegir de un pequeño menú de opciones, la métrica que será utilizada y según la opción elegida, se aplican las reglas de negocio para tal caso.

3.4.3 Diagrama de Clases.

En el diagrama de clases que podemos observar en la Figura 3, nos muestra la planeación de la estructura lógica que se pretende que tenga la herramienta. Como puede observarse en cada módulo existe apoyo para el usuario a manera de Archivos de Ayuda y además de ofrecer una impresión del resultado de los cálculos, es posible guardar copia del reporte.

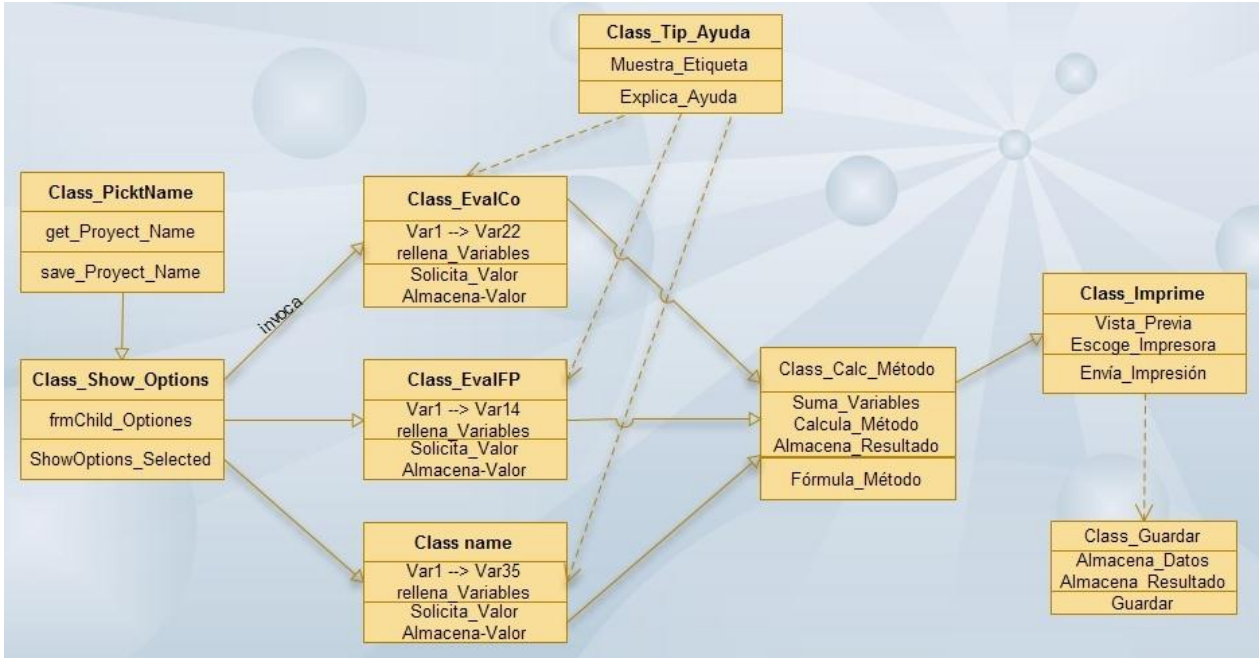


Figura 3.- Diagrama de Clases de la Aplicación

3.4.4 Diagrama de Actividades.

Nos muestra en una línea del tiempo los momentos en que el usuario interactúa con el sistema y los eventos que ocurren en cada intervención de ambos, hasta el momento en que interviene el periférico "impresora" y finaliza un ciclo normal en el uso del programa. (Figura 4)

3.4.5 Diagrama de Entidad Relación.

Es posible examinar la relación entre los elementos de la Base de datos, (Figura 5) que aunque en el análisis de requerimientos se determinó que no era del todo necesario utilizar una Base de Datos, es posible implementarla aprovechando la mínima dificultad que ésta tiene.

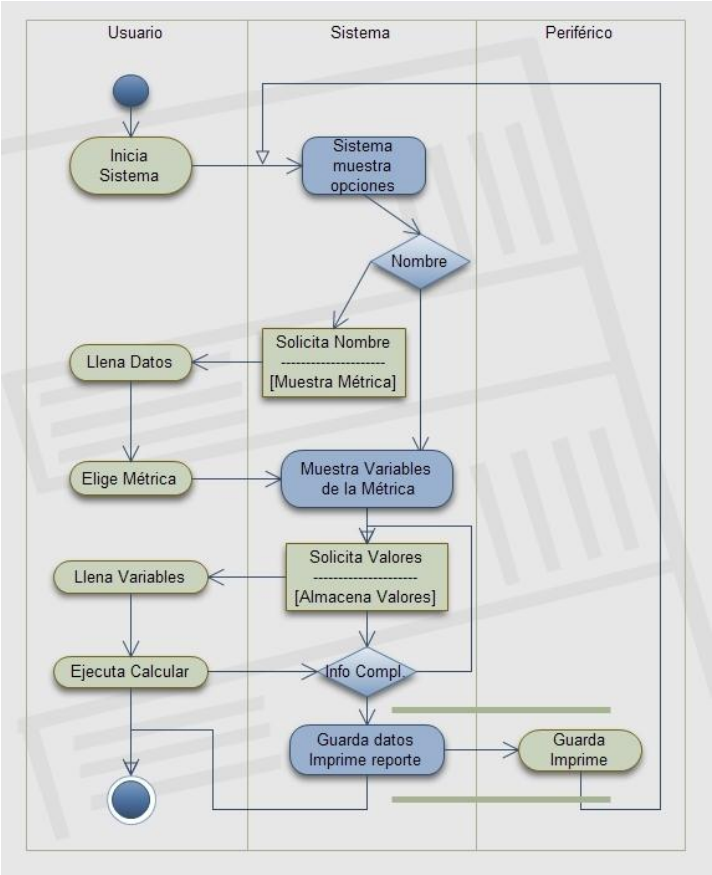


Figura 4. Diagrama de Actividades del Sistema

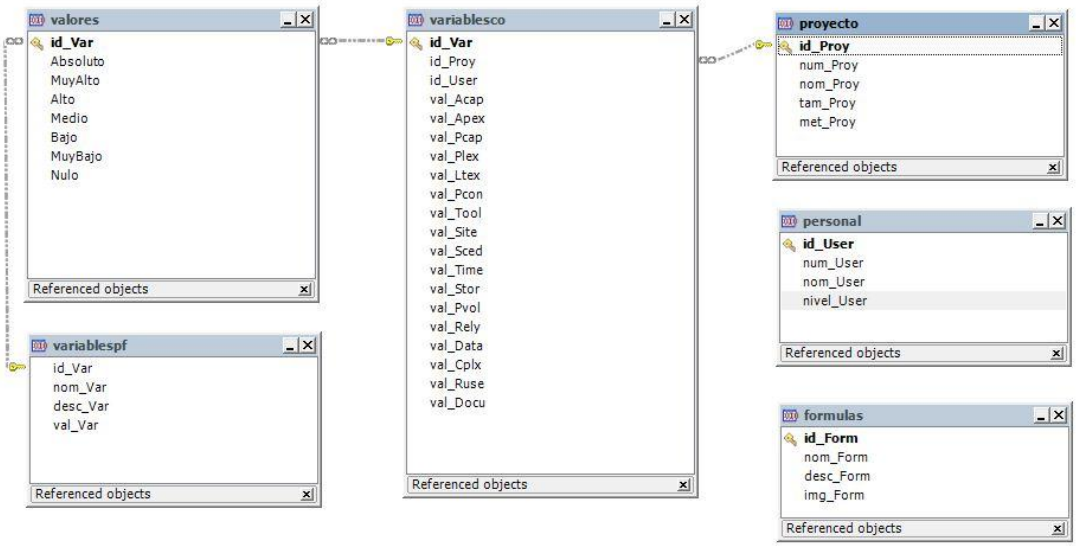


Figura 5. Diagrama Entidad Relación de la Base de Datos del Sistema

3.5 Metodología.

Como se ha planteado en los requerimientos del sistema, se habrá de diseñar diferentes vistas para la interfaz con el usuario a fin de facilitar el acceso e interpretación de los datos. Se preparará una aplicación básica de escritorio del tipo multi – documento, en la que se contengan las vistas necesaria para cada una de las opciones de interacción con éstas.

3.5.1 Diagrama de Flujo del Sistema.

Como podemos observar en la figura 6, una secuencia lógica pero muy general del flujo de la aplicación estaría determinada por los siguientes puntos:

1. Mostrar una ventana de inicio o “Splash_Win”, como común mente se les conoce.
2. Validar o registrar Usuario.
3. Ofrecer entrada de datos, ya sea por consola o puntero (Teclado o ratón).
4. Después de elegir una metodología, recibir información del usuario, para cada variable.
5. Realizar los cálculos correspondientes (calcular la estimación)
6. Emitir un reporte, preguntando al usuario si se requiere además impreso.
7. Si el usuario lo solicita, guarda la información en la Base de Datos y ofrece nueva tarea o Finaliza el Programa.

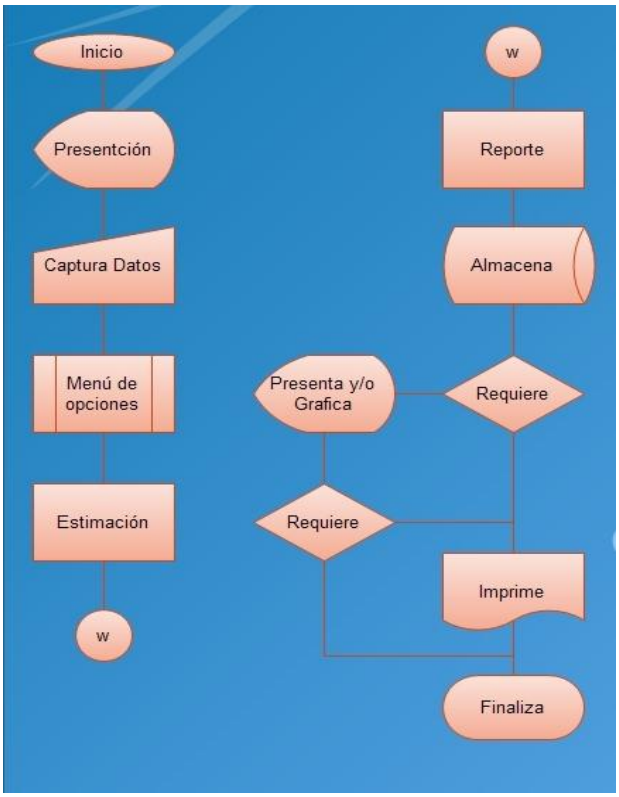


Figura 6. Diagrama de flujo general del sistema.

Los cálculos a realizar en el paso cuatro tendrán que ver con cada una de las metodologías utilizadas en el programa, el símbolo de proceso “Estimación” contiene los algoritmos que dan solución a las tres tareas principales que se proyecta que hará el programa y que son:

1. Estimar el tamaño
2. Estimar el esfuerzo
3. Determinar su Duración

A partir de estas evaluaciones, será posible entregar al usuario un reporte medianamente completo, acerca de los costos que derivarán del proyecto que pretende realizar y en una etapa posterior podrá obtener en ese mismo reporte, una aproximación de los recursos necesarios para cumplimentarlo en un determinado tiempo posible.

3.5.2 Estimar el tamaño.

Para determinar el tamaño de una aplicación de software, utilizaremos una métrica establecida y probada que hemos comentado en el segundo capítulo y que se conoce como el Análisis de Puntos de Función (Function Point Analysis, FPA). El FPA tiene como propósito solucionar algunos de los problemas asociados con el cálculo del tamaño del software medido en Líneas de Código (Lines of Code, LOC).

3.5.3 Estimar el Esfuerzo.

Otro método que esta aplicación utiliza para realizar una estimación, es COCOMO II (comentado ampliamente en el capítulo dos), que entre algunas de sus características nos ayuda a calcular las conocidas variables LOC: (líneas de código) que es una de las más habituales y antiguas. Además de DSI (instrucciones de código fuente entregadas), con sus modificadores o múltiplos “K” de miles.

3.5.4 Determinar su duración.

A partir de los cálculos que determinan el tamaño que podría tener el desarrollo planteado, así como el esfuerzo que habría de emplearse, se podrá determinar a modo de estimación, cuál sería la duración del proyecto o visto de otro modo se estaría en posibilidad de proponer un costo para dicho proyecto.

3.6 Resumen.

Básicamente se está planteando el esquema general que podría tener la aplicación y la secuencia lógica de sus elementos. Encontramos a través de esquemas y diagramas, explicación más visual de todos los componentes que contendrá en sí el programa y cómo se conjugarán para dar fluidez a una interacción con el usuario, así por lo tanto, encontramos también una explicación gráfica de los momentos en que todos estos elementos participan.

También se ha planeado el flujo de las funciones básicas que se espera se tengan del programa, que son recabar datos, calcular y estimar tanto el tamaño como el esfuerzo requerido, para así, calcular y proponer una duración de un proyecto de software.

Capítulo IV
Implementación.

4.1 Introducción.

Todo desarrollo de ingeniería posee la peculiaridad de estar debidamente pensado, desde el punto de vista de habérsele aplicado ingenio. Es decir, atravesó por un riguroso esquema de programación que involucró desde un planteamiento, el análisis de todas sus partes, el diseño de los elementos o subsistemas, hasta la implementación de éstos para poder comprobar lo planeado y obtener así algún resultado.

La Ingeniería de Software prevé que se aplique esta misma planificación a todo desarrollo de software y llegar a un resultado positivamente tangible, si es que la creación de software permitiera tal situación, solo después de haber recorrido cada una de sus etapas de manera exitosa. Esto es, haber tenido primordialmente una etapa de análisis y diseño bien definida, antes de pasar a la fase de codificación e implementación

Habiendo logrado un buen análisis de los requerimientos y seleccionado el ambiente de trabajo idóneo será posible iniciar la etapa del desarrollo o que conocemos también con el nombre de implementación, misma que se refiere a proponer el o los algoritmos necesarios para la automatización y a la codificación de dichos programas. El mejor plan para llevar al cabo dicha automatización, deberá incluir al menos los siguientes aspectos.

- Definir el cronograma de implantación.
- Dar a conocer el plan o estrategia a seguir al supervisor, líder de equipo, o implicados.
- Determinar necesidades y o requerimientos del plan elaborado
- El plan debe ser actualizado y ejecutarse en el período de tiempo más breve posible.
- Poner en práctica lo establecido en la documentación.
- Recopilar evidencia documentada de los avances.

4.2 Plan de Desarrollo.

Buscando un soporte adecuado a la Justificación y enfoque del presente proyecto, se realizó un cuestionario tipo encuesta y se envió vía correo electrónico a 31 empresas mexicanas del ramo del desarrollo de software. Solo 12 de ellas tuvieron la amabilidad de contestar de manera positiva y ayudar así a la investigación.

Esta encuesta se practicó con la finalidad de obtener opiniones por parte de los encuestados, a cerca de su interés en conocer o hacer uso de las métricas de software. El cuestionario se preparó con los tópicos suficientes para recabar la información requerida y con algunos otros prácticamente de relleno. La tabulación de la información recibida, se practicó en una hoja electrónica, se preparó una tabla de datos con formato condicional, donde se ponderó de acuerdo con las respuestas obtenidas una tabulación muy sencilla, pero que nos permitió establecer un panorama más preciso de las costumbres técnico-administrativas que tiene la industria de software en México.

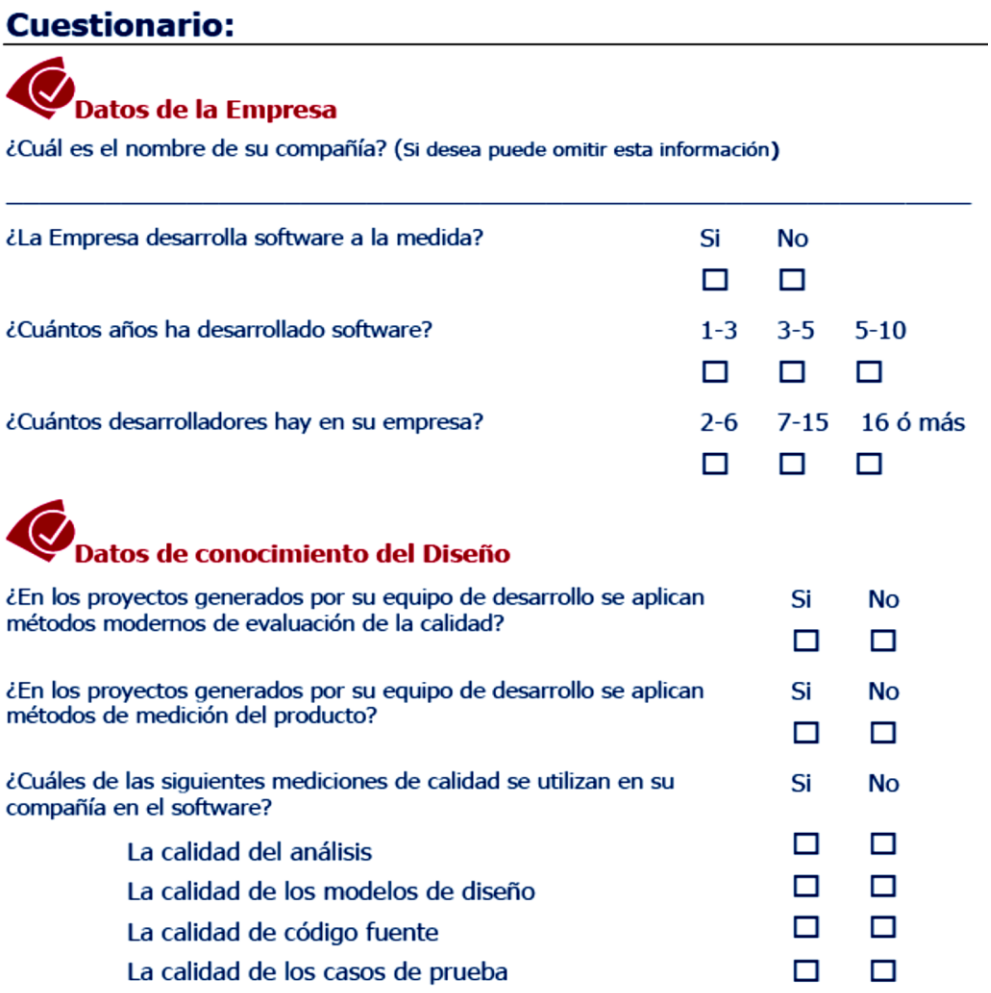
Tratando de ser un poco analíticos y tomando en cuenta que las empresas que respondieron, en algunos de los casos son trasnacionales, (Tabla 8) lo que significa que sus métodos y procedimientos pueden estar regulados en otro país, sería alarmante aunque no descabellado afirmar, que en otras partes del mundo, tampoco serían muy alentadoras sus respuestas en cuanto a sus procedimientos de planificación de proyectos.

Tabla 8. Listado de empresas encuestadas.

Nombre de la Empresa	Dirección de Contacto	Resp. Negativas		Resp. Positivas	
Gedas north america	ventas@cosmos.com.mx	10	63%	6	38%
Softek	webmaster@softtek.com	8	50%	8	50%
Hores mexico	atencionclientes@horesmexico.com	8	50%	6	38%
Abits, software	ventas@abits.com	4	25%	12	75%
Skavoovie web services	contacto@skavoovie.com.mx	4	25%	11	69%
Microsiga Mexico	jbuitron@microsiga.com	6	38%	5	31%
Infonexo	info@consultorialatina.com.mx	5	31%	11	69%
Ort media lab	contacto@ortmedialab.com	7	44%	9	56%
Microsys	ventas@microsyssolutions.com	3	19%	13	81%
Softelligence	clientes@softelligence.com.mx	3	19%	10	63%
Software-shop	ventas@software-shop.com	2	13%	14	88%
Webcom Technology	ventas@webcomtechnology.com.mx	5	31%	10	63%

En la Tabla 8, podemos observar también la dirección de contacto de cada empresa y un reporte general de la cantidad de respuestas positivas y negativas que de cada caso se recibieron. Es necesario insistir, en que no todas las solicitudes fueron atendidas y que aún de las atendidas, algunas de las respuestas no tuvieron una respuesta ya que según argumentaban algunos de los encuestados, que se abstenían por razones de seguridad o por políticas de la empresa.

De acuerdo con el tipo de preguntas contenidas en el cuestionario (Figura 7), podríamos hacer un podo de estadística en diversos rubros, pero la motivación de la encuesta estaba específicamente dirigida a encontrar una restricción o por el contrario un aliciente de la utilidad de construir la herramienta prototipo que trata este documento.



Cuestionario:

Datos de la Empresa

¿Cuál es el nombre de su compañía? (Si desea puede omitir esta información)

¿La Empresa desarrolla software a la medida?	Si ☐	No ☐	
¿Cuántos años ha desarrollado software?	1-3 ☐	3-5 ☐	5-10 ☐
¿Cuántos desarrolladores hay en su empresa?	2-6 ☐	7-15 ☐	16 ó más ☐

Datos de conocimiento del Diseño

¿En los proyectos generados por su equipo de desarrollo se aplican métodos modernos de evaluación de la calidad?	Si ☐	No ☐
¿En los proyectos generados por su equipo de desarrollo se aplican métodos de medición del producto?	Si ☐	No ☐
¿Cuáles de las siguientes mediciones de calidad se utilizan en su compañía en el software?	Si	No
La calidad del análisis	☐	☐
La calidad de los modelos de diseño	☐	☐
La calidad de código fuente	☐	☐
La calidad de los casos de prueba	☐	☐

Figura 7. Fragmento del cuestionario practicado en la Encuesta.

Nota: En los Anexos de este documento se encuentra una copia del cuestionario completo.

4.3 Análisis de las Métricas.

Consiste en analizar la manera en la que los procedimientos llevarían un mejor trayecto y desenlace de tal forma que se optimicen los resultados tanto en tiempo como en calidad del proceso, brindando la posibilidad de implementar la mejora y de ser posible estandarizar, para después lograr una automatización. Este tipo de tarea corresponde al ejemplo que se muestra en la Figura 6. El seguimiento adecuado de una metodología, nos llevará a contar con una Métrica como tal.

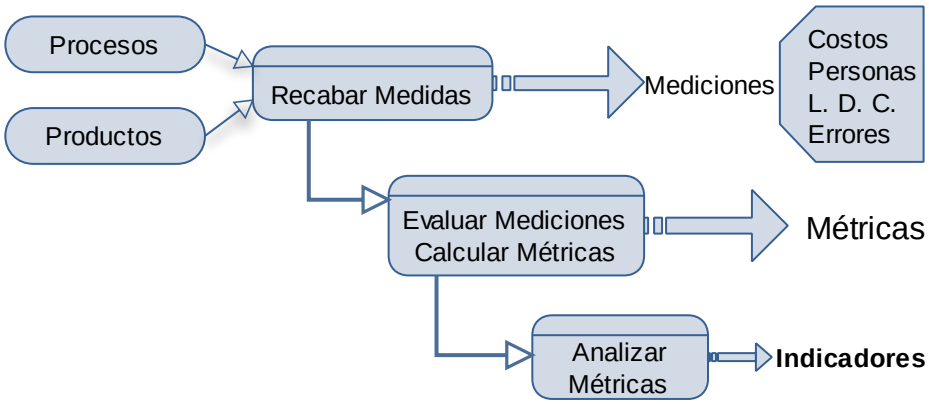


Figura 8. Diagrama del planteamiento y análisis de una Métrica.

4.3.1 Analizando COCOMO II.

La herramienta propuesta, servirá como un apoyo al Ingeniero en Sistemas (I. S.) en el momento de presupuestar algún proyecto de software que esté planificando o que vaya a realizar. Por lo que en ella se manejarán lo que conocemos como metodología IFPUG (International Function Point Users Group) y COCOMO (Constructive Cost Model), ambas son metodologías empleadas para realizar estimaciones de los tiempos y esfuerzos que representara la elaboración de determinado trabajo de desarrollo de software El conjunto de las variables de la primera metodología están establecidas así como la métrica en sí, desde principio de la década de los ochenta y aunque se ha ido actualizando, básicamente se refieren a las mismas áreas de evaluación, que son las siguientes:

Formula:

$$E = a(Kl)b * m(x)$$

donde:
"E" es Salario/mes (Media).
"a" y "b" son constantes según el modo (Orgánico, Semilibre o Rígido) (Tabla 9).
"Kl" es la cantidad de líneas de código (En miles).
"m(X)" es el multiplicador que depende de 15 atributos constantes. [4]

Tabla 9. Ponderación estándar de los factores de Complejidad Técnica.

MODO	a	b	c	d
Orgánico	3.20	1.05	2.50	0.38
Semilibre	3.00	1.12	2.50	0.35
Rígido	2.80	1.20	2.50	0.32

Atributos:

Cada atributo se cuantifica para un entorno de proyecto. El factor de escala es muy bajo - bajo - nominal - alto - muy alto - extremadamente alto. Dependiendo de la calificación de cada atributo, se asigna un valor para usar de multiplicador en la fórmula (por ejemplo, si para un proyecto el atributo DATA es calificado como muy alto, el resultado de la fórmula debe ser multiplicado por 1000). El significado de los atributos según su tipo, se explica para cada variable en la Tabla 10.

Tabla 10. Variables COCOMO y su descripción por grupo de correspondencia.

De Software	
RELY:	Fiabilidad o garantía de funcionamiento requerida al software. Indica las posibles consecuencias para el usuario en el caso que existan defectos en el producto. Va desde la sola inconveniencia de corregir un fallo (muy bajo) hasta la posible pérdida de vidas humanas (extremadamente alto, software de alta criticidad).
DATA	Se refiere al tamaño de la base de datos en relación con el tamaño del programa. El valor del modificador se define por la relación: D / K, donde D corresponde al tamaño de la base de datos en bytes y K es el tamaño del programa en cantidad de líneas de código.
CPLX	Representa la complejidad del producto que habrá de realizarse, en comparación con otros que el mismo equipo de trabajo.
De Hardware	
TIME	Posibles limitaciones en el porcentaje del uso de la CPU. (considerando cantidad de equipos vs. Programadores.
STOR	Posibles limitaciones en el porcentaje del tiempo de acceso o uso de la memoria disponible.
VIRT	El grado de volatilidad de la máquina virtual. (Variable que debe considerarse de acuerdo con la plataforma elegida para el desarrollo.
TURN	Tiempos de respuesta requeridos para las “corridas” o sesiones de ejecución.
De Personal	
ACAP	Capacidad de los Analistas que participan en el desarrollo.
AEXP	Experiencia en aplicaciones similares de los Analistas integrantes del equipo que participan en el desarrollo.
PCAP	Capacidad de los Programadores, para trabajar en equipo y en el manejo de la plataforma elegida.
VEXP	Experiencia del personal en el manejo del tipo de máquina virtual que se utilizará.
LEXP	Experiencia de los participantes en el uso del lenguaje de programación que se va a utilizar.
De Proyecto	
MODP	Empleo de prácticas modernas de programación.
TOOL	Empleo de herramientas sofisticadas para el desarrollo de software
SITE	Capacidad del equipo de trabajo de trabajar remotamente (multi-site)
SCED	Grado de necesidad de la planeación requerida.

4.3.2 Analizando Puntos Función.

En la segunda fase de la herramienta se propone utilizar la métrica FPA (Function Points Analysis ó Análisis de los Puntos Función Los Puntos Función (FP) son una unidad de medida lógica de las funciones del sistema de software desde la visión del usuario. Éstos proveen el valor esencial de lo que es el software y qué hace con los datos desde el punto de vista del usuario. Su potencia proviene del énfasis sobre el punto de vista externo. Debido a que la estimación de esfuerzo y costo basada en FPA no depende del lenguaje y tecnología utilizados, se puede disponer del cálculo desde las primeras etapas del desarrollo, particularmente los FP pueden calcularse desde la etapa de análisis de requerimientos.

Las métricas orientadas al tamaño son objeto de polémicas y no están aceptadas universalmente como el mejor modo de medir el proceso de desarrollo de software. La mayor parte de la discusión gira en torno al uso de las SLOC como medida clave. Sus defensores afirman que es un artificio que se puede calcular fácilmente para todos los proyectos de desarrollo de software, que muchos modelos de estimación del software existentes las utilizan como elemento de entrada y que existe un amplio conjunto de datos y de literatura basados en SLOC. En el extremo opuesto, los detractores afirman que las medidas en SLOC son dependientes del lenguaje de programación, que perjudica a los programas más cortos pero bien diseñados, que no se pueden adaptar fácilmente a lenguajes no procedimentales y que su utilización en la estimación requiere un nivel de detalle que puede ser difícil de conseguir [5].

Muchos ingenieros de software prefieren estimar la funcionalidad en vez del tamaño físico. El concepto de funcionalidad captura la noción de cantidad de función contenida en un producto entregado o en la descripción de lo que se supone será el producto [6]. Las métricas orientadas a la función son medidas indirectas del software y del proceso por el cual se desarrolla. Estas métricas se centran en la "funcionalidad" o "utilidad" del programa [5].

Para sustentar este enfoque de funcionalidad se han desarrollado métodos que han sido probados y revisados sobre la base de la experiencia. Estos métodos se conocen como Métodos de Análisis de Puntos Función. El principal y más grande organismo defensor de estos métodos es la IFPUG, (explicado en el capítulo dos). Sin embargo una cuestión muy importante es el ámbito de aplicación de las métricas. La medida de FPAs se diseñó para ser utilizada en aplicaciones de sistemas de información principalmente considerados de gestión.

Con este enfoque, los componentes de un sistema están representados por cinco clases denominadas características generales del sistema: entradas externas (External Inputs), salidas externas (External Outputs) y consultas externas (External Inquires), [22] cada una de las cuales interactúa con archivos y son llamadas transacciones. Los otros dos son los archivos lógicos internos (Internal Logical Files) y los archivos de interfaz externa hacia otras aplicaciones (External Interface Files) (Figura 9)

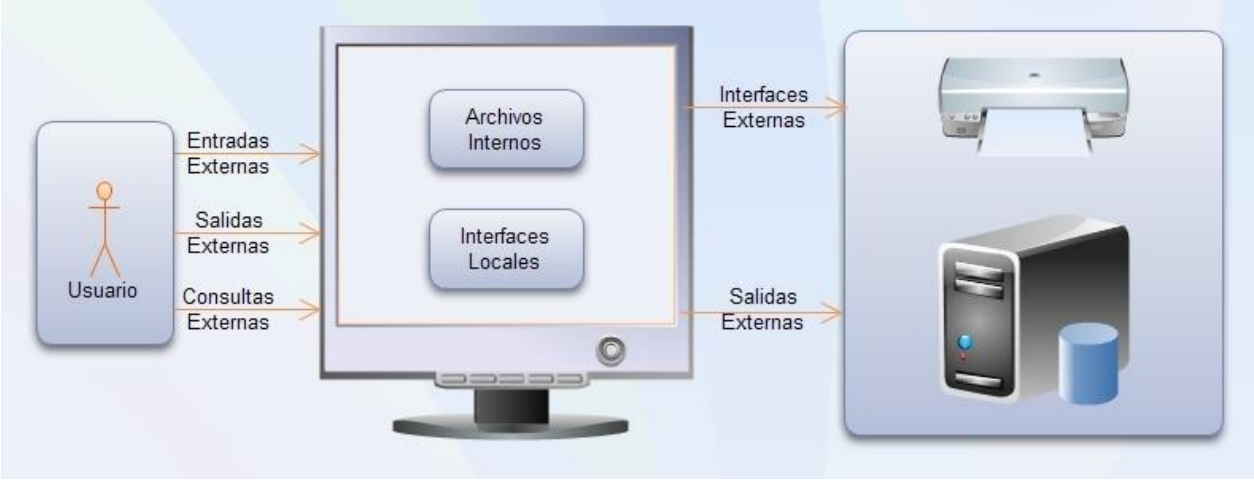


Figura 9. Modelo de Puntos Función.

Debido a que los FP miden los sistemas desde una perspectiva funcional, son independientes de la tecnología. Independientemente del lenguaje, método de desarrollo o plataforma de hardware usados, la cantidad de FP permanecerá constante. La única variable es la cantidad de esfuerzo necesario para entregar un conjunto de FP. Por lo tanto, el FPA se puede usar para determinar si una herramienta, un entorno o un lenguaje es más productivo comparado con otros, dentro de una organización o entre organizaciones.

Entradas externas (EI) es un proceso elemental en el cual los datos cruzan el límite desde el exterior hacia el interior del sistema. Estos datos pueden venir desde una pantalla de entrada de datos o desde otra aplicación. Los datos pueden ser usados para mantener uno o más archivos lógicos internos.

Salidas externas (EO) es un proceso elemental en el cual los datos derivados atraviesan el límite desde el interior hacia el exterior del sistema. Un EO puede actualizar un archivo lógico interno. Los datos crean reportes o archivos de salida hacia otras aplicaciones. Esos reportes y archivos son creados a partir de uno o más archivos lógicos internos y archivos de interfaz externa.

Consultas externas (EQ) es un proceso elemental con componentes de entrada y salida que resulta en el retorno de datos desde uno o más archivos lógicos internos y archivos de interfaces externas. El proceso de entrada no actualiza archivos ILF y la salida no contiene datos derivados.

Archivos lógicos internos (ILF) es un grupo de datos lógicamente relacionados, identificable por el usuario, que residen completamente dentro del límite de la aplicación y es mantenido a través de las entradas externas.

Archivos de interfaz externa (EIF) es un grupo de datos lógicamente relacionados, identificable por el usuario, que se usa solamente con propósitos de referencia. Los datos residen completamente fuera de la aplicación y son mantenidos por otra aplicación, o incluso por un hardware especial.

Una vez que todos los componentes han sido clasificados como EI, EO, EQ, ILF o EIF, se les asigna una posición en la escala que puede ser: bajo, medio o alto (Low, Average or High). Para las transacciones (EI, EO, EQ) esa posición se basa en la cantidad de archivos actualizados o referenciados (FTR) y la cantidad de tipos de elemento dato (Data element type - DET). Un DET es un campo único, no recursivo, reconocible por el usuario. En el caso particular de las EQ se ubican en la escala (bajo, medio o alto) como EO, pero se les asigna un factor como EI. Si el mismo FTR se usa en la parte de entrada y salida se cuenta una sola vez. Si el mismo DET se usa en la parte de entrada y salida, también se cuenta solo una vez.

Vista así, esta métrica considera:

- El Tamaño: observa el tamaño, no la calidad con la que se realizó el software, o el valor que finalmente podría representar.
- Aplicaciones: mide las aplicaciones de software, sin considerar el hardware que se emplea en su realización, ni la administración del proyecto, ni la documentación, etc.
- Funcionalidad: se refiere a la capacidad del software para que un usuario pueda realizar transacciones (lectura, escritura, etc.) y el guardar datos.
- Usuario: será quien lo va a usar y no quien lo desarrolló o quien lo diseñó. Podría verse a esta métrica como el “metro”, para medir el tamaño de una aplicación de software o del valor de ese producto, o del esfuerzo requerido para desarrollarlo, etc.

Este Modelo se rige por el siguiente esquema:

1. Determinar el tipo de conteo
2. Identificar los alcances de la medición y los límites de la aplicación.
3. Contar las funciones de datos, identificar y contar la capacidad de almacenamiento de los datos.
4. Contar las funciones transaccionales. La capacidad de realizar operaciones.
5. Determinar los puntos de función no ajustados: sumar el número de componentes de cada tipo conforme a la complejidad asignada.
6. Determinar el valor del factor de ajuste.
7. Determinar los puntos función ajustados. Se consideran los puntos función no ajustados

La IFPUG, el grupo internacional de usuarios de puntos función publicó un caso de estudio que ilustra la aplicación de la FPA IFPUG proponiendo reglas para un análisis orientado a los objetos y al diseño. Este caso de estudio utiliza modelos orientados a objetos en los que los métodos de las clases son idénticos a las funciones especificadas en el documento de requisitos. Bajo este supuesto, los métodos o clases, pueden considerarse directamente como transacciones. La información general de esta propuesta es que no existe documentada una validación del procedimiento ni tampoco una herramienta que lo soporte. Tampoco existen mayores datos o publicación del modelo ya sea empírico o metodológico.

Básicamente propone que el éxito de la metodología depende de si la medición se realiza en la fase de análisis o construcción. En la fase de análisis, pueden utilizarse los requerimientos de los usuarios, diagramas de clase genérica o diagramas de casos de uso. Por otra parte, en la fase de construcción, ha de basarse en los requisitos de usuario actualizados, en los diagramas de clases orientadas a objetos o diagramas de casos de uso, así como las ventanas o 'vistas' de interfaz gráfica de usuario.

Esto además de apoyar una teoría de mejora basada en la aplicación correcta de los mecanismos de evaluación y análisis para lograr una correcta estimación en tiempo de diseño, refuerza la hipótesis de la necesidad de una herramienta de software capaz de coadyuvar en esta etapa del proceso de presupuesto de un proyecto, con la finalidad de acercar al Ingeniero de Software a un estadio de certidumbre en el momento de diseñar su contrato de servicios.

4.4 Garantizar facilidad de acceso.

Al ser éste un único desarrollo, una de las principales prioridades, será en todo momento la facilidad de uso del programa ya que el objetivo es convertirla en una herramienta de apoyo no de obstáculo para el I.S. Así, al ofrecer formularios con opciones apoyado en los famosos "check box" o los conocidos "option box" elementos básicos de ventanas en casi todo lenguaje de programación de aplicaciones para Windows[R]

4.5 Datos de la Planificación.

La planificación incluye desde luego información diversa que se estructura en una clasificación de la siguiente manera:

- Configurar el Entorno de Trabajo.
- Diseñar e Implementar la Base de Datos: (sentencias SQL: create table... etc.)
- Definir herramientas, bibliotecas, puestos de trabajo, etc.
- Diseñar y estructurar los componentes.
- Codificar y Documentar los componentes.
- Realizar pruebas unitarias o por componentes.
- Verificas si los componentes interactúan correctamente a través de sus interfaces, cubren la funcionalidad establecida y los requisitos no funcionales (pruebas de integración)
- Verificar la integración del sistema globalmente, las interfaces de los distintos subsistemas que lo componen y con el resto de SI con los que se comunica (prueba del sistema)
- Se puede definir un plan de pruebas incremental.

4.6 Configuración del entorno de trabajo.

Considerando la posibilidad de brindar un proyecto de acceso libre y que pueda ser utilizado por cualquier persona, independientemente de la plataforma que utilice o de sus preferencias por algún tipo de software, se propone utilizar una base de datos montada en MySQL (Figura 10), debido a que es un Manejador de Datos de libre acceso y de amplia difusión. Sin embargo también existe una versión demostración del gestor de Bases de Datos de Microsoft[R], el MS-SQL Server. Para éste desarrollo se preparó un script de MySQL que Anexa en la sección correspondiente.

Sin embargo al confeccionar la herramienta se utilizó para fines prácticos una versión de la Base de Datos montada en SQL Server debido a la transparencia con que maneja los accesos a datos desde el IDE de Ms Visual Basic® que con el fin de evitar redundancia y contenido innecesario, de esta versión no se incluirá la definición script.

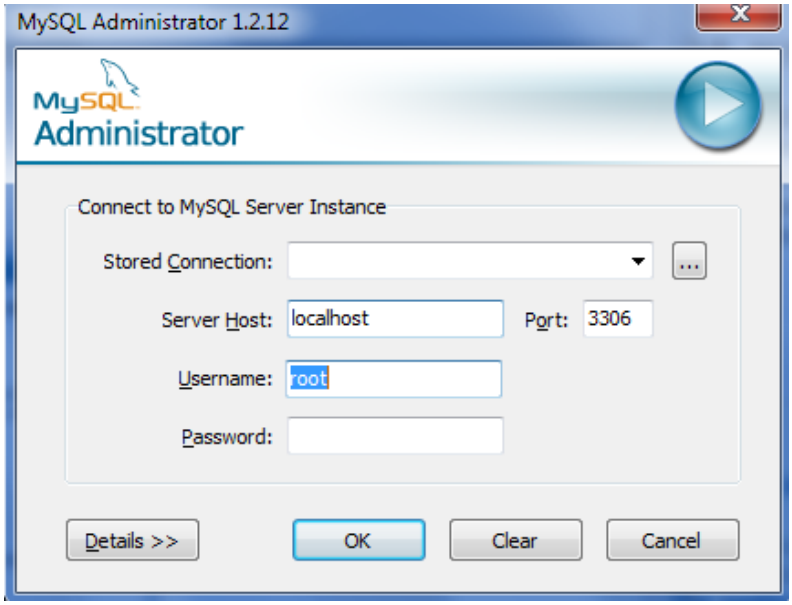


Figura 10.- Imagen de inicio del Gestor de BD de acceso libre.

Del mismo modo se proyectó utilizar el IDE de NetBeans, que es considerado uno de los más versátiles y completos entornos de programación para el lenguaje Java. Este entorno maneja integralmente la API de java y permite una programación sencilla además de ofrecer ayuda para el programador, resultando de éste aplicaciones de índole profesional. (Figura 11)

Figura 11.- Vista del SplashScreen del IDE NetBeans Versión 6.9

El IDE de NetBeans, al ser de acceso libre también, tiene la capacidad de incorporar accesorios de los llamados “Plug-Ins” de terceros y con ello ampliar de manera considerable sus opciones gráficas y/o de acceso a datos. Tal es el caso de *JasperReports* o *ZK Direct* que provee de una interfaz gráfica tipo web muy moderna y fácil de emplear. Sin embargo, y como se comentó desde la planeación, para éste desarrollo se ha decidido utilizar el IDE de MS-VisualBasic, (Figura 12.), por razones principalmente de tiempo, debido a la factibilidad de diseño de vistas en dicho IDE.

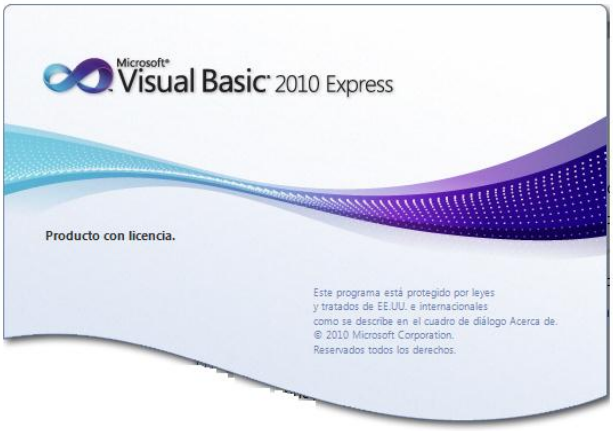


Figura 12.- Vista del SplashScreen del IDE de MS-Visual Basic 2010.

Como una opción adicional y de mejor acceso a los elementos de la base de datos, se utilizará la herramienta TOAD, que también es de acceso libre y contiene opciones de visualización de datos y algunas parecidas a un ETL. El TOAD (Tools for Oracle Application Development) Figura 13, es básicamente un conjunto de herramientas usadas para desarrollar y asistir a los desarrolladores en aplicaciones basadas en Oracle y servicios web en la plataforma Windows, que también está disponible para otras bases de datos como Microsoft SQL Server, IBM DB2 y MySQL.

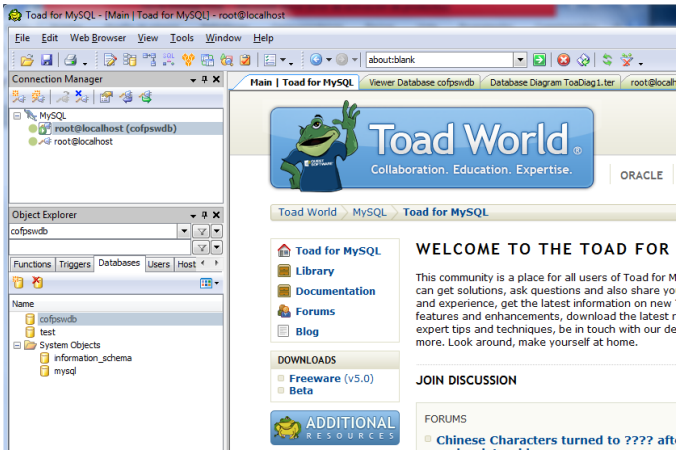


Figura 13.- Vista del entorno bienvenida de TOAD

ETL son las siglas en inglés de Extraer, Transformar y Cargar (Extract, Transform and Load). Es en sí un proceso de manejo de la información que permite a una entidad u organización acceder y manipular datos desde múltiples fuentes, limpiarlos, a veces transformarlos y cargarlos en otra base de datos, data mart, o data warehouse para analizar, o en otro sistema operacional para apoyar un proceso de negocio.

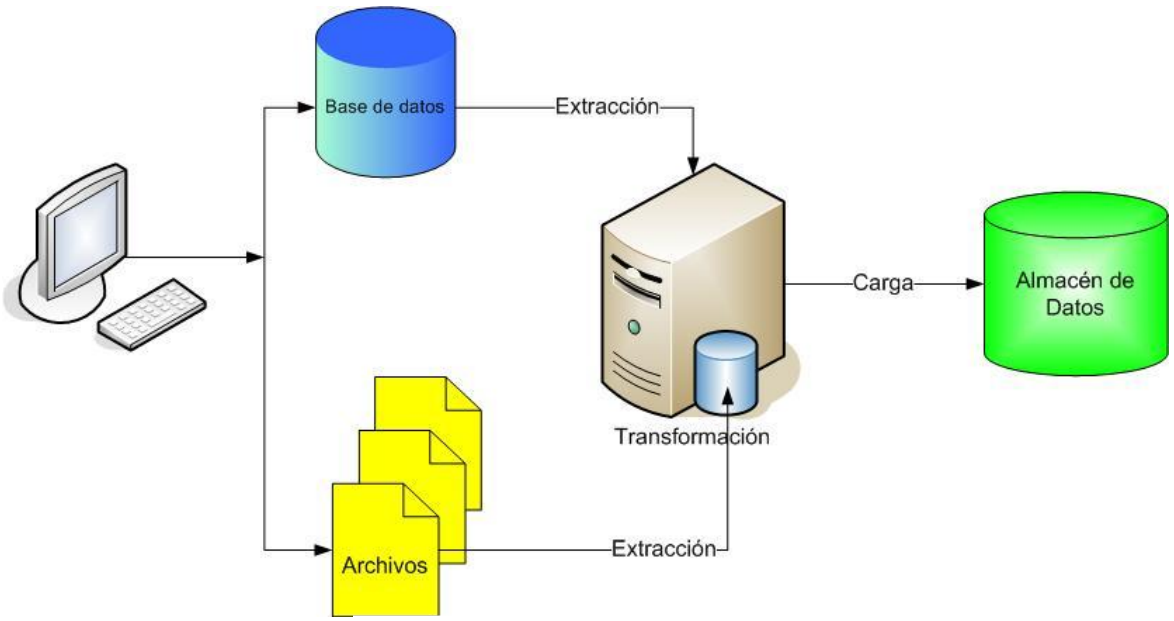


Figura 14.- Esquema típico de una herramienta

En un esquema Típico de cualquier herramienta ETL, como el de la Figura 14, deberían observase de forma general, las siguientes funcionalidades:

- Control de la extracción de los datos y su automatización, disminuyendo el tiempo empleado en el descubrimiento de procesos no documentados, minimizando el margen de error y permitiendo mayor flexibilidad.
- Acceso a diferentes tecnologías, haciendo un uso efectivo del hardware, software, datos y recursos humanos existentes.
- Proporcionar gestión integrada del Data Warehouse y los Data Marts existentes, integrando la extracción, transformación y carga para la construcción del DataWareHouse corporativo y de los Data Marts.
- Uso de la arquitectura de metadatos, facilitando la definición de los objetos de negocio y las reglas de consolidación.
- Acceso a una gran variedad de fuentes de datos diferentes.
- Manejo de excepciones.
- Planificación, logs, interfaces a schedulers de terceros, que nos permitirán llevan una gestión de la planificación de todos los procesos necesarios para la carga del DW.
- Interfaz independiente de hardware.
- Soporte en la explotación del Data Warehouse.

Los procesos ETL también se pueden utilizar para la integración con sistemas heredados.

4.6.1 Identificación de componentes funcionales

La técnica de modelado funcional (análisis funcional / funcional descomposición) se utiliza para modelar la relación entre las transacciones y la aplicación como un todo. Las transacciones se asignan a una jerarquía funcional de la aplicación en la actividad a la que contribuyen. Se utiliza los siguientes criterios, siempre que sea posible, para comprobar cada tarea para determinar si es un único proceso elemental (transacción de lógica de negocios). Se cuenta como una única transacción lógica cuando:

- tiene lógica de procesamiento (edición, validación, etc.) diferente de otras operaciones similares,
- que tenga acceso a una combinación única de campos y archivos,
- al finalizar, deja el negocio en un estado consistente y predecible,
- es usuario reconocible y definible,
- está creado por los requerimientos del negocio y no los requisitos técnicos de la solución elegida,
- es lógicamente independiente de otras transacciones (aunque se puede, en algunos casos se activa por otra transacción),
- es lógicamente activados por un evento externo,
- logra un objetivo de negocio, no un objetivo técnico

4.7 Diseño de la Base de Datos

Según el estado del Arte, cuando se ha requerido sistematizar herramientas para esta tarea específica, no ha sido necesario dotar a la herramienta de una Base de Datos, puesto que solo se realizan cálculos muy sencillos para una computadora. Pero de acuerdo con el modelo que se propone en este trabajo, dichos cálculos se basarán en aplicar estadística a los antecedentes del mismo equipo de trabajo, por lo que serán más precisos y podrá ofrecerse al usuario de la herramienta un nivel de sugerencia más acertado, desde el punto de vista estadístico.

De tal forma que lo que se requerirá conservar en dicha base será, por un lado cada uno de los proyectos realizados por el mismo equipo de trabajo. Tomando en cuenta que el líder de proyecto o Ingeniero de Software, será el mismo y que siempre trabaja al menos con el mismo número de elementos hardware-software-humanwere, y por otro, el valor registrado para cada una de las variables que sirven de soporte para determinar las evaluaciones de costos.

Para ejemplificar mejor la estructura que guarda la base, la figura 7 muestra un esquema de la Base de Datos en su etapa preliminar.

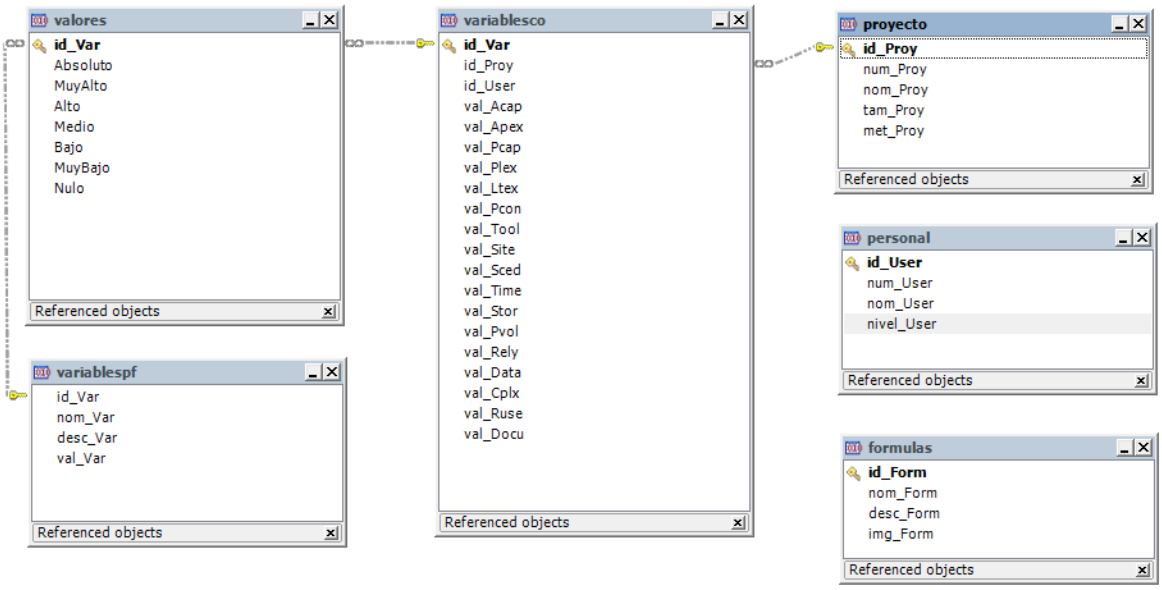


Figura 15. Diagrama de la estructura de la Base de Datos.

Con la finalidad de cumplir con el objetivo de facilitar el empleo de la herramienta que resultará de este trabajo, se ha diseñado un script .sql que crea la estructura inicial de la Base de Datos. A pesar de que un solo administrador de bases de datos como lo es el TOAD es suficiente para ejecutar las sentencias estándar del lenguaje estructurado para consultas, existen detalles en el Sql nativo que en ocasiones lo hacen parecer incompatible. Como se ha comentado en el punto de la Configuración del Entorno de Trabajo, Se prepararán los dos modelos de base de datos, meramente por compatibilidad entre los programas elegido, Pero a pesar de que se entregará una carpeta con todos los archivos que componen el desarrollo, para evitar redundancia y contenido innecesario, en la parte de anexos de este documento, solo se encontrará el listado script de la base diseñada en MySQL.

Como opciones alternativas de programación y tomando en cuenta que es un desarrollo básicamente experimental y que lo que se está buscando es lograr un entorno útil y accesible para cualquier usuario, también se han considerado otros IDEs tanto para el lenguaje Java, como el C++ y el Visual Basic. Tal es el caso del entorno de Helios conocido como “eclipse”, el cual permite a su vez la integración de sockets de terceros como en NetBeans. Con ello, fue posible integrar la herramienta ZK Direct que provee de una interfaz gráfica tipo web muy moderna y fácil de emplear.

4.8 Sistema a Desarrollar.

Básicamente se presentará en una aplicación MDI (Multi Document Interface) muy sencilla que ofrezca la posibilidad de elaborar planes de presupuesto en diferentes proporciones, es decir, en el que se pueda elegir entre diferentes modos o planes, para que brinde la posibilidad al I. S. de variar entre opciones, según sus necesidades y pueda realizar sus presupuestos o estimaciones considerando diferentes escenarios o alternativas.

El sistema contará con una presentación, un cuerpo MDI con Barra de herramientas, Barra de menús, controles típicos de escape y ayuda para el Usuario, En una de sus opciones del menú Archivo, se colocará la opción de "Nueva Estimación", lo que llevará en automático a un formulario multi—tab (figura 8) en el que podrá seleccionar paso a paso, las opciones que se requiere considerar para realizar dicha estimación.

Como en cualquier aplicación de escritorio al iniciar el programa, aparece una vista de presentación, comúnmente llamada SplashWin o SplashScreen, mostrado en la Figura 16, en la que se puede apreciar el logo o distintivo del software y la información básica como la versión del producto y el famoso "copy right". El ejemplo de la ilustración fue confeccionado en VisualBasic y NetBeans.

Figura 16.- Vista del SplashScreen de la herramienta Cofpsw.

Después de algunos segundos aparece la ventana de la herramienta, que básicamente es una aplicación MDI (Multi Document Interface) que significa que contendrá dentro de sí, el resto de las ventanas que componen el software. Esta versión preliminar cuenta con una barra de menús básica complementada con algunos atajos en botones de una típica barra de herramientas, como se puede observar en la Figura 17.

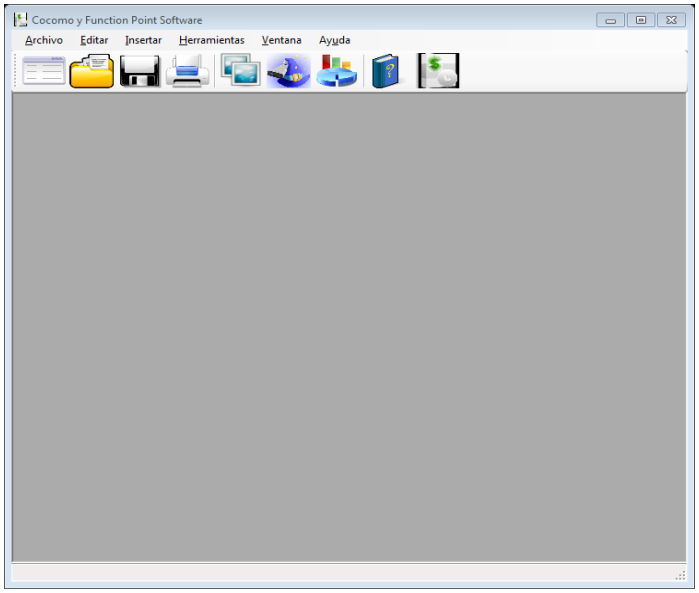


Figura 17.- Vista del entorno MDI de la Herrmienta Cofpsw.

Al iniciar el programa aparecerá la ventana de opciones, que se refiere a las métricas que podemos aplicar al nuevo proyecto de software, para poder evaluar su costo. Dichas opciones son las que se ha comentado al través del documento, Cocomo II, Ifpug y una tercera compuesta por una fusión de ambas. Antes de ver los botones que permiten elegir de entre ellas, aparece una ventana explicativa que aclara el objeto de los botones, sus selecciones y la necesidad de especificar un nombre para el proyecto en cuestión. Al Elegir "siguiente" tenemos la vista de la Figura 18.

Figura 18.- Vista de la ventana de opciones de la Herramienta Cofpsw.

Al elegir el botón de la opción COCOMO II, aparecerá la ventana de captura de los datos requeridos por el sistema para poder calcular el posible costo en tiempo (meses—hombre) de acuerdo con esta métrica y estar en posibilidad de ofrecer alguna sugerencia y fuera el caso, como vemos en la Figura 19.

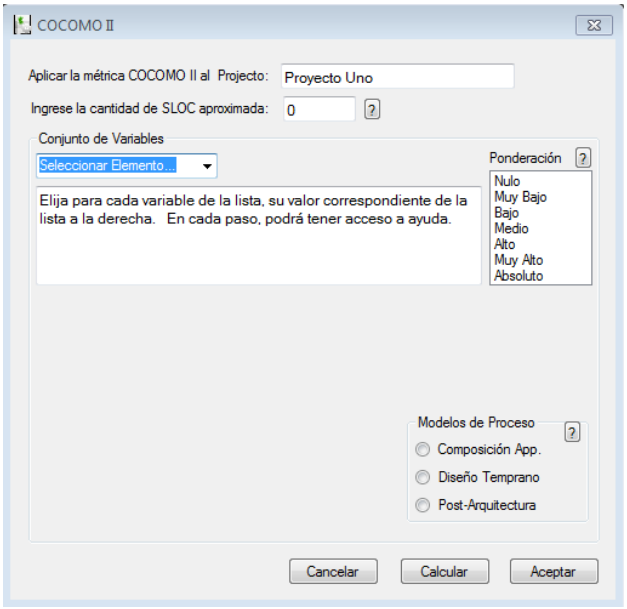


Figura 19.- Vista del módulo 1, que calcula COCOMO II

En cuanto aparece ésta vista, debe proporcionarse toda la información que la metodología requiere para poder realizar los cálculos correspondientes, como por ejemplo debe ingresarse el número aproximado de líneas de código fuente que se estima habrá de contener el proyecto. Por el contrario la métrica de IFPUG, automatizada en el segundo módulo de la herramienta, no necesita el dato de SLOC, porque precisamente es uno de los puntos ventaja de esta metodología. Tal como se aprecia en la Figura 20, ese valor resulta de evaluar el nivel de complejidad que presentan los elementos a construir durante el desarrollo del proyecto en cuestión.

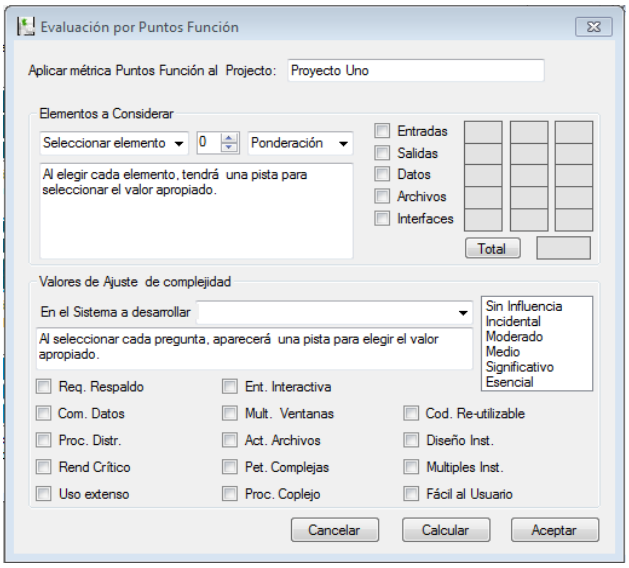


Figura 20.- Vista del módulo 2 que calcula el método de IFPUG

No obstante, la segunda fase del módulo, contiene tal vez menos detalle de la información que recaba del usuario, para determinar el nivel de esfuerzo necesario a invertir, para que el equipo de desarrollo lleve a término el proyecto en evaluación.

4.8.1 Lista de Cambios.

En futuras revisiones aparecería en este apartado, todo lo relacionado con los cambios y las causas detalladas que los originen. Dichos cambios son considerados según su relevancia una nueva versión del producto. En esta versión que aunque podría ser solo preliminar, se presentan los alcances de acuerdo con los objetivos planteados, pero dejando abierta la posibilidad de complementarlo o practicar adición de ciertas funciones que lo mejoren o sofistiquen.

Tabla 11. Lista de Cambios a la versión.

Rubro	Fecha de Inicio	Motivo del Cambio
Aplicación	Agosto de 2010	Inicio del Proyecto
Versión	1.0.0	Aplicación de nueva creación
Componente	Splash -Win. MDI –Win Options –Win Help –Win About –Win Reports –Win Tools –Win View –Win	
Responsible	El Autor	Durante este proyecto no habrá más colaboradores.

4.8.2 Restricciones encontradas.

Debido a que no se emplearon conceptos o tecnologías nuevas de software, realmente no hubo restricciones técnicas para el desarrollo de la herramienta. Sin embargo al querer buscar que el resultado sea una aplicación multiplataforma, obliga a que el desarrollo se efectúe en un lenguaje de alta portabilidad, como lo es el java y que los enlaces a la base de datos así como la misma Base estén construidos en un software gestor de bases de datos de acceso libre como lo es MySQL.

Será posible vincular herramientas de éstos tipos con los plug-ins adecuados, o como comúnmente se conocen, los “sockets” o conectores para enlazarlos. En éste caso se utilizó el conector para bases de datos abiertas conocido como ODBC, Figura 21, que permite al programador que dentro de código en Java, pueda manejar una API, que manipula el acceso a datos en una BD de MySQL.

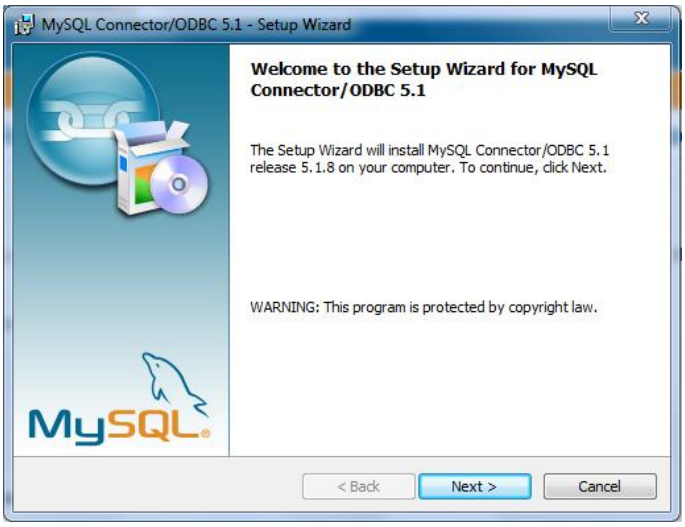


Figura 21. Vista de la ventana de instalación del socket ODBC

4.9 Resumen.

La implementación de un software necesariamente conlleva la utilización de diversas herramientas, muchas de las cuales ahora son muy sofisticadas y permiten una gran flexibilidad en el trabajo, proporcionando al programador un sin número de opciones. Se pretende así que el presente desarrollo se convierta en una de esas tantas herramientas de apoyo al desarrollador de software, que le permita optimizar su trabajo y apoyarle en el cumplimiento de sus metas.

En el presente capítulo se explica el diseño y la implementación del trabajo propuesto, pero no se aclara suficientemente, tal vez, la forma de operación de la herramienta software que integra CocomoII y Function Points (Por lo que se denomina CoFpSw), ya terminada. Para tal efecto, se anexa un Manual de Usuario de la aplicación, en la sección de Anexos de éste documento.

Capítulo V
Pruebas y Resultados.

5.1 Introducción.

Algunos autores reconocidos en materia macro y micro económica, que han tratado el tema de la estimación de costos coinciden en la necesidad y utilidad de esta práctica, DeMarco por ejemplo, [14] propone dos definiciones sucesivas muy realistas de lo que es una estimación, por un lado la idea implícita de que una estimación es la predicción más optimista que tiene una probabilidad no nula de llegar a ser cierta. Lo que nos entrega a un amplio margen a suspicacias, por otro lado nos dice que una estimación es una predicción que tiene la misma probabilidad de estar por encima que de estar por debajo del resultado real. Lo que suena a auténtico azar. La única manera segura de poder tener unos estándares de productividad buenos y dignos de ser utilizados es no depender de lo que dicen libros y artículos (con datos obtenidos en condiciones a menudo bien diferentes), sino de disponer de los datos de productividad propios, que pueden haber sido obtenidos en proyectos anteriores del mismo tipo (en cuanto a aplicación y tecnología) y con equipos de desarrollo de calificaciones y características personales conocidos.

Todo ello es la justificación a este trabajo, de haber propuesto que las estimaciones se calculasen con base en resultados estadísticos emanados de los registros de los trabajos previos realizados por un mismo equipo de desarrollo y que prevén para un mismo desarrollador, o ingeniero de software, una estimación más acertada o más cercana a la realidad.

Como producto final se alcanzará una herramienta de software capaz de auxiliar a cualquier ingeniero de software a soportar de manera confiable sus presupuestos, pero que será aún más eficiente, en la medida en que se retroalimente con el mismo uso y que mantenga el mismo equipo y estilo de trabajo. Sin esperar que suceda lo contrario si cambia de personal, pero serán más precisas sus estimaciones si mantiene el perfil de éstos, en cuanto a capacidad, conocimientos y experiencia en el manejo de las herramientas que utilizan.

5.2 Equipo de Cómputo.

Para el presente desarrollo, no será necesario adquirir equipo especial ni tampoco hacer inversiones en Software, toda vez que el sistema o herramienta se desarrolla como parte del proyecto de esta tesis. Por lo tanto, se empleará el equipo de cómputo destinado para tal efecto. En la Tabla 7 se establecen las características de todo el sistema de información que se empleará. La mayor parte del software es de acceso libre, o cuenta con licencia de uso de software preinstalado. En apoyo a este sistema se cuenta también con un equipo portátil con un sistema operativo Mac OS 10.5 en el que también se encuentra instalado el software de programación empleado, que es de acceso libre y a excepción también del que es exclusivamente para el sistema operativo Microsoft Windows. El detalle de todas estas características es posible observarlo en la Tabla 12.

Tabla 12. Descripción del Equipo de cómputo y atributos de software.

Hardware disponible:	Software Instalado:
→ Procesador AMD AthIon™ II X2 a 2.8 GHz. → Memoria RAM DDR 4 GB → HDD SATA (Capacidad formateado) 320 GB. → DVD SATA Grabador SuperMulti. ATAPI → Pantalla 19 " ViewSonic. → Adaptador Gráfico: NVIDIA™ GeForce. 6150 → Modem V.90 internacional. → Ethernet LAN. NVIDIA nForce 10/100 mbps Scanner HP 3670 Impresora HP Laser 1020 Memoria USB 4 GB Sony	→ Microsoft® Windows® 7 Ultimate Edition → Microsoft® Office® 2007 Enterprise → NetBeans IDE 6.8 (Oracle SUN) → COSTAR de Sofstar.com → MySQL 5.3 → Abbyy System → Acrobat Reader → ArgoUML 0.32 → Nero Essentials 8 → Corel Draw X5 → E-DRAW MAX 4.5

5.3 Esquemas utilizados como ejemplo.

Una herramienta de este mismo tipo o muy similar, se intentó confeccionar por estudiantes de la Universidad de Alcalá [41], quienes intentaron automatizar también las tareas que por separado realizan las métricas de “COCOMO II” y “Function Points” (Ilustraciones 12 y 13) utilizando formularios y controles de Microsoft VisualBasic 6.0, pero quienes a pesar de haber planteado teóricament lo que en una etapa

final podría solucionar el tener ésta herramienta, no pudieron publicar resultados contundentes referentes a la eficiencia o secuencia que guardara dicho programa.

Otro caso de Ejemplo es la herramienta que automatiza el Modelo COCOMO II exclusivamente y que es la que confecciona el mismo organismo que investiga y propone la metodología desde sus inicios, que está a cargo del Dr. Boehm en la Universidad del sur de California[38][39]. Dicha herramienta está disponible en una versión reducida para su evaluación en el sitio de la escuela y que no obstante que es un software probadamente útil, exige al usuario un conocimiento no superficial tanto de la métrica como de la nomenclatura que el mismo establece, haciéndola difícil de manejar y contradictoriamente inútil como una herramienta de apoyo si no se domina el tema.

5.3.1 Diagramas del primer ejemplo

En la Figura 22 se puede observar, el intento por combinar los elementos necesarios para realizar una evaluación utilizando la métrica de COCOMO II, con un control de comando, que aparentemente mostrará alguna otra vista con manejo de datos referentes a la métrica de Puntos Función. Por otro lado, en la Figura 23 (que debido a su ilegibilidad por el tamaño, reaparecerá en el anexo D), se muestra una especie de tabulación, en la que se puede seleccionar el valor para cada una de las variables necesarias para el cálculo de los parámetros, según dicha metodología.

Figura 22. Vista del esquema de ejemplo “Ventana de captura de datos para evaluación COCOMO”.

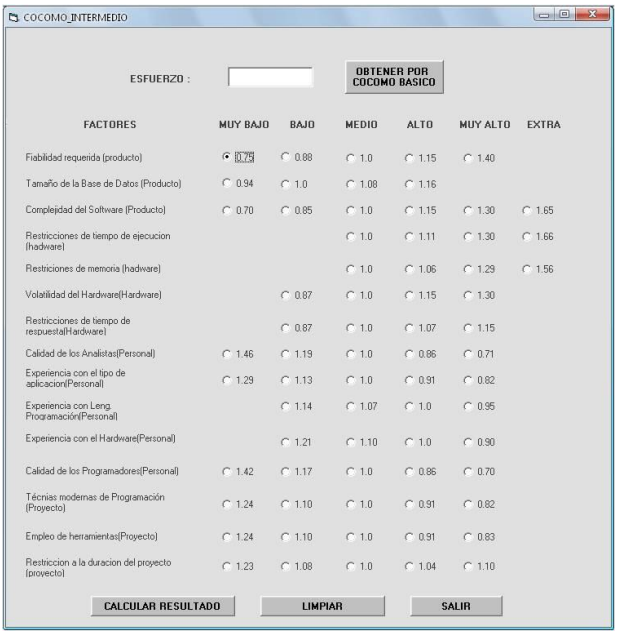


Figura 23.- Vista del esquema de ejemplo: “Tabla de selección de valores para cada factor”.

5.3.2 Diagramas del segundo ejemplo.

En el caso de la herramienta de USC (University of Southern California), evidentemente es, hasta cierto punto muy completa y ha probado ser muy útil; empero, no presenta la información de manera clara o entendible, (Figura 24) a menos que el Ingeniero de Software que la esté utilizando, sea versado en la metodología y en su forma de aplicación.

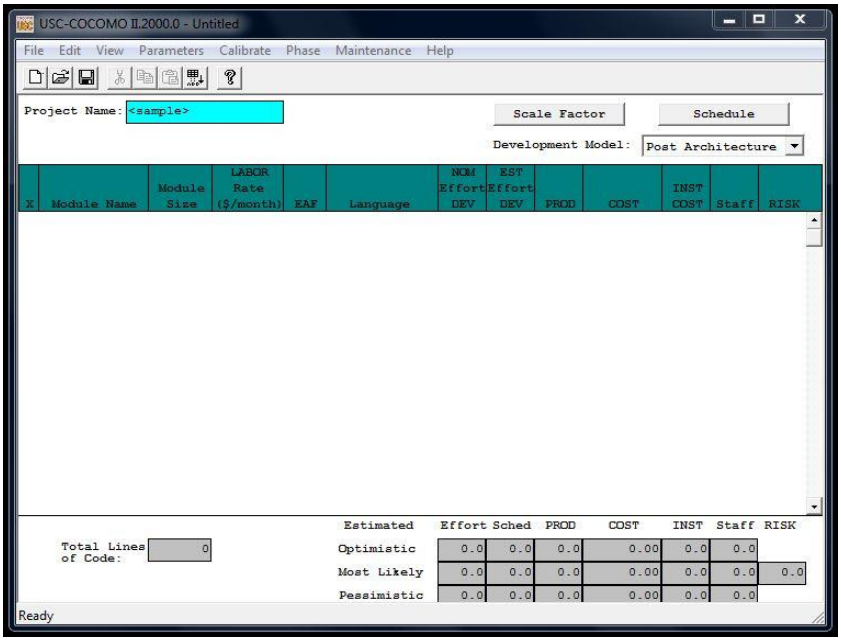


Figura 24.- Vista del entorno de USC COCOMO.

5.4 Propuesta COFPSW.

Considerando la versatilidad de cada una de las metodologías analizadas y después de haber evaluado diversas alternativas, esta tesis plantea una manera diferente de obtener una estimación a partir de la información que proporciona el usuario, tal como lo hacen las métricas COCOMO y Function Points. Sin intensión de proponer una nueva metodología[2], se aborda la posibilidad de aprovechar en parte cada una de ellas dentro de la misma herramienta de software.

La mayor virtud dentro de la metodología de evaluación por Puntos Función, es quizás la habilidad de calcular los componentes del desarrollo que está siendo planificado, a partir de una ponderación con base en la complejidad de los mismos. Por otra parte la habilidad que tiene el método de COCOMO de obtener un estimado del esfuerzo que ha de emplearse, para llevar al cabo un proyecto, está cifrado en una cantidad más alta de variables, lo que presupone mayor detalle en el cálculo de los factores que intervienen.

5.4.1 Modelando el Planteamiento.

Considerando lo anterior, la figura 25 muestra gráficamente lo que propone Cofpsw, es básicamente un esquema que toma la parte de la métrica Puntos Función que obtiene el SLOC, que de acuerdo con lo comentado en la sección 4.3.2 dicha metodología evalúa la complejidad de las funciones que contiene un proyecto. Por otro lado, se toma de la métrica Cocomo II, la parte que cuantifica los conjuntos de atributos que cualifican cada aspecto del equipo de programadores involucrado en el desarrollo, tal

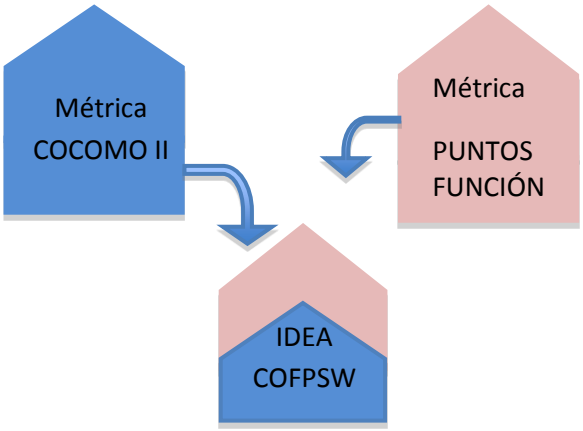


Figura 25. Diagrama de la estructura de Cofpsw

[2] La creación o planteamiento de una nueva metodología es un trabajo completo vinculado con la investigación y el método científico. Se requiere cubrir rigurosos aspectos y cubrir otro tipo de objetivos que no son los planteados en el presente trabajo.

como es referido en el análisis del apartado 4.3.1. En ambos casos, se explica a detalle cual es la esencia de cada uno de los métodos, para acercarse a una interpretación básicamente financiera de los aspectos involucrados en el desarrollo de software.

Es decir, en la herramienta propuesta, existen tres módulos de evaluación de proyectos de software que nos proporcionan un estimado de los costos que representara un determinado proyecto. Uno de ellos se apega a lo planteado por la metodología de Boehm, el segundo aborda el análisis según la aportación de Albretch. En el tercer módulo se hace un acopio de ambos métodos para obtener una estimación, con lo que se podría obtener una optimización de ambos.

5.4.2 Optimizando las Estimaciones.

De acuerdo con el análisis del planteamiento de COCOMO II, la ponderación que se utiliza para elegir emana de una serie de proyectos que fueron estudiados y minuciosamente registrados, considerando todos los aspectos que intervinieron en su diseño, configuración y realización. El estudio sobre estos proyectos, articuló cuidadosamente una ponderación que finalmente se establece como un estándar de medición para cualquier otro proyecto de software, siempre que aquel se haya planificado en todos los aspectos.

A partir de ello, continuamente se ha demostrado el grado de certidumbre que ofrece una estimación de costos en las tres principales etapas de un proyecto de desarrollo de software, realizada con esta metodología. Por otro lado, el análisis que se realizó sobre la forma en que IFPUG realiza las estimaciones del mismo tipo, muestra que en efecto se basa en una tabulación de valores preestablecidos para llevar a término esas evaluaciones.

En adición a ello, la idea de Cofpsw es que para cada desarrollador o ingeniero de software que utilice la herramienta, tenga una opción adicional y que en algún momento se sirva de ésta para tener estimaciones cada vez más certeras, fundamentadas en un registro histórico de sus propios desarrollos, valiéndose para ello, de almacenar en la Base de Datos del sistema cada una de las evaluaciones que haya practicado y que en adelante esa información se convierta en una compilación que no solo apoye sus evaluaciones al servirle de referente sino que a su vez, actúe como una información estadística haciendo más precisos sus cálculos.

5.4.3 Aplicando la Optimización.

Dentro del tercer módulo de la herramienta, se encuentra una vista que muestra la automatización de los elementos propuestos por la evaluación Cofpsw como puede apreciarse en la Figura 26. Este módulo combina los elementos de las dos metodologías anteriores, pero además de ofrecer una mejora sobre el método del módulo 1 al no solicitar un estimado de SLOC, sino que auxilia al usuario a obtenerlo de manera precisa, al calcularlo indicándole el grado de complejidad de las funciones que contendrá el proyecto. Como lo efectúan los otros módulos en la aplicación, Cofpsw almacenará los niveles de dicha complejidad, en cada uno de los factores que intervienen en la realización del proyecto evaluado.

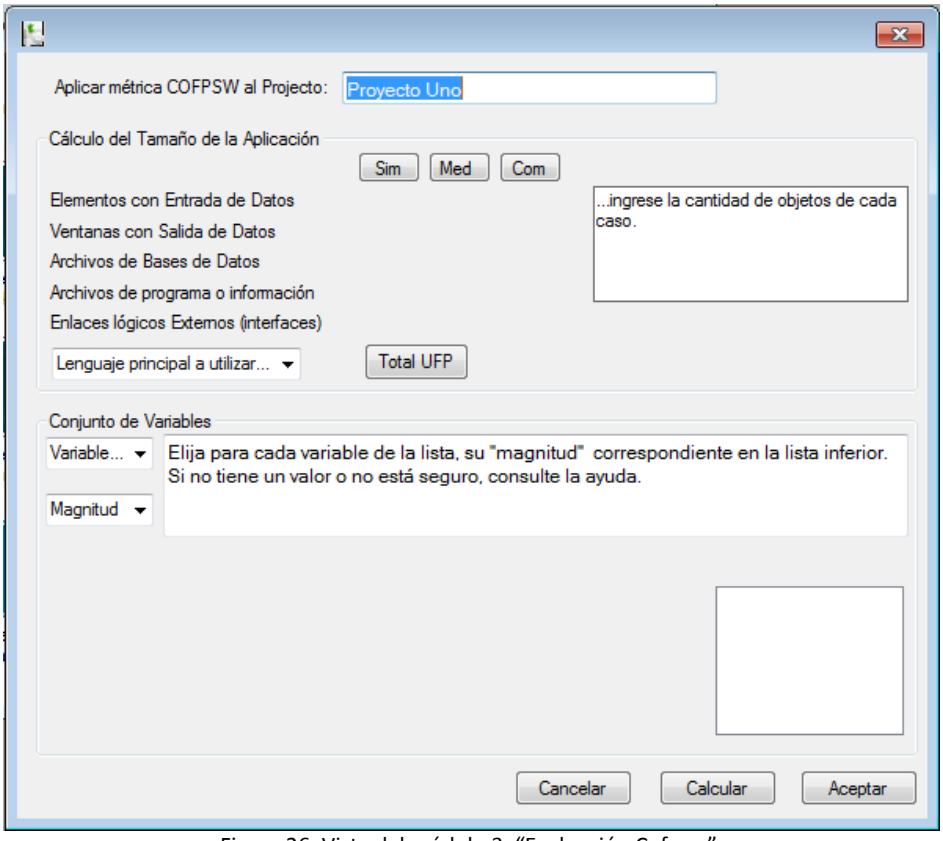


Figura 26. Vista del módulo 3, “Evaluación Cofpsw”.

Al almacenar dichos valores, estará preparando al mismo sistema, para realizar en un futuro próximo, valoraciones estadísticas y dar la opción al usuario de utilizarlas en sus próximas valoraciones presupuestarias. Es decir, a elección del Ingeniero de Software, encargado de la planificación de un proyecto de software y para quien una de sus actividades es realizar un estimación del costo total que tendrá realizar dicho proyecto, para estar en posibilidades de brindar un presupuesto lo más acertado posible, de lo que conllevará económicamente hablando, la realización del mismo.

5.4.4 Especificando los elementos de la optimización.

La manera en que el módulo Cofpsw trabaja con los datos es hasta cierto punto convencional, si hablamos de un sistema de información que se basa en el registro de sus operaciones en una Base de Datos como tal, para mantener un histórico y después poder aplicar operaciones de recuperación y cálculo en ese histórico.

Al llevar un registro de cada uno de los proyectos que presupuesta o evalúa el mismo ingeniero de software, que está trabajando con el mismo equipo de analistas y programadores, que se vale además del mismo equipo de cómputo y utilizando las mismas herramientas de software, que si acaso realiza algunos cambios para mejorar en cuanto a sus procesos; La información recabada de cada uno de sus anteriores trabajos, servirá como un apoyo estadístico, para evaluar un nuevo proyecto.

La intención es que cuando un Ingeniero de Software, requiera evaluar un nuevo proyecto, con el fin de hacer una estimación y se valga de esta herramienta, tenga como alternativas aplicar la metodología de Cocomo II, la metodología de IFPUG o la de Cofpsw indistintamente. La ventaja que ofrecerá la opción Cofpsw, como ilustra la Figura 27, es que podrá omitir el ingreso de algunos datos pudiendo elegir en su lugar, que el programa ofrezca un dato alternativo, que obtendrá de un promedio de los datos que estén almacenados en la Base de Datos para el mismo elemento.

Figura 27.- Esquema de recuperación de los datos almacenados

5.5 Pruebas y Resultados.

5.5.1 Fase de Pruebas.

Al someter a evaluación un proyecto de software, en el que se desarrollaría un sistema de información que serviría para llevar un control de los pacientes que ingresan al área de cuidados intensivos en el hospital #72 del IMSS, resultó ser similar en cuanto a complejidad, a otros desarrollos de software realizados por la misma empresa de software. Por lo tanto el ingeniero de software a cargo del desarrollo, decide practicar una evaluación de costos valiéndose exclusivamente del tercer módulo de la herramienta Cofpsw.

Durante el proceso, se decide hacer uso de la opción de "Parametrización Estadística" de este módulo, misma que provoca que cada uno de los valores que deberían ser proporcionados por el usuario de la herramienta, sean calculados internamente y sugeridos mediante un cálculo estadístico del histórico de valores almacenados en cada una de las variables de datos que conforman la tabla de datos del sistema.

Ejemplo:

Uno de los elementos cruciales en la estimación de costo de un proyecto de ésta índole, es la experiencia que tienen los programadores miembros del equipo, con la terminología médica que se empleara en el diseño de las vistas. Pues al ser éste un requerimiento funcional delicado y esencial por tratarse de una aplicación de este tipo. Así mismo la variable de fiabilidad del sistema (RELY),[9] que implica la relevancia de los errores de programación, al tratarse de un sistema en el que se registrarán pacientes de un hospital; se convierte en un elemento de peso para la solución del problema.

Después de haber utilizado la herramienta Cofpsw en seis desarrollos previos, ésta contiene un conjunto de valores diversos para cada factor de influencia. Si particularmente se presta atención que para la primero de ellos, almacenado con la variable "*RELY*" y que está almacenada en el campo de datos denominado "*Val_Rely*" (Figura 28) , se observa que para dicho factor se cuenta con un histórico de H=[1.4, 1.15, 1.4, 1.4, 1.55 y 1.4] que puede interpretarse como una clara costumbre en éste equipo de desarrollo, de participar en proyectos de alto nivel de riesgo en la responsabilidad de la programación de las aplicaciones en las que ha participado.

nom_Proyecto	Val_Prec	Val_Flex	Val_Resl	Val_Team	Val_Rely	Val_Data
Proyecto1	1.400	0.500	0.500	0.500	0.500	0.450
Facilidades	1.150	0.350	0.650	0.700	0.750	0.650
Prueba3	1.400	0.900	1.000	1.100	1.000	0.900
Proyect4	1.400	0.100	1.300	1.600	1.000	0.900
Niveles	1.550	0.750	0.950	0.900	1.000	0.100
Prototipo1	1.400	1.140	1.000	0.850	0.900	1.280

Figura 28.- Vista de la Tabla "Proyectos" de la Base de Datos del Sistema, (Fragmento).

Aunque esto puede derivarse de muchas razones, lo que atañe a ésta prueba es que la opción de "Parametrización Estadística" con que cuenta el módulo tres de la herramienta, realiza una media aritmética a los datos registrados en el histórico de cada variable o driver de coste que maneja. Esto es: en el caso de la variable "*RELY*" con los valores H=[1.4, 1.15, 1.4, 1.4, 1.55, 1.4] y aplicando (4) tenemos:

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n} \qquad (4)$$

$$\mu(H) = \frac{(1.4 + 1.15 + 1.4 + 1.4 + 1.55 + 1.4)}{\sum i} = \frac{8.3}{6}$$

$$\mu(H) = 1.3833$$

Esto significa que el programa realiza una asignación de promedio extraído de todos los valores que han sido almacenados en el campo de la misma variable, según lo hace el fragmento de código que aparece a continuación. Guarda en la variable "Curr_Prec" el valor resultante de la consulta SQL con promedio que en éste caso es de 1.38. En seguida, hay un control que evitaría que el valor resultante, estuviese muy por encima de la media y que pudiera arrojar valores inesperados.

```
Curr_Prec = SELECT Val_Prec FROM VariablesCo WERE id_Proyecto =
CurrPro
If (Curr_Prec – Prom_Prec ) < 1 then
    Curr_Prec = Prom_Prec
Return Curr_Prec
```

La idea básica, es que en determinado momento el programa sea capaz de apoyar al ingeniero de software a realizar sus valoraciones, aún que él no ingrese el valor preciso de todos los factores y no necesariamente se calculen con los valores nominales ya establecidos de las metodologías analizadas, sino que genere un valor a partir de los datos que el mismo equipo de trabajo ha ido generando.

5.5.2 Etapa de Resultados.

Cuando se confrontaron los resultados emitidos por cada una de las métricas de evaluación que automatiza la herramienta propuesta, pudo observarse que como en toda apreciación presupuestal que se pretenda realizar sobre elementos en los que intervienen diversos factores, incluidos los humanos, siempre existe cierto nivel de incertidumbre.

Específicamente en el proyecto evaluado, del sistema de registro de pacientes, los resultados en cada uno de los módulos resultaron muy cercanos entre sí pero, a pesar de que el proyecto se completó sin contratiempos y en condiciones normales, los resultados que arrojó en términos reales, tuvieron algunas diferencias.

Como es posible apreciar en la gráfica de la figura 29, el módulo tres, representado por las barras en color más claro (rótulo Cofpsw), logró resultados de la estimación un tanto alejados de los resultados arrojados por las métricas establecidas, pero en términos prácticos fue el que tuvo la menor diferencia con respecto de los resultados que reportó el proyecto real terminado.

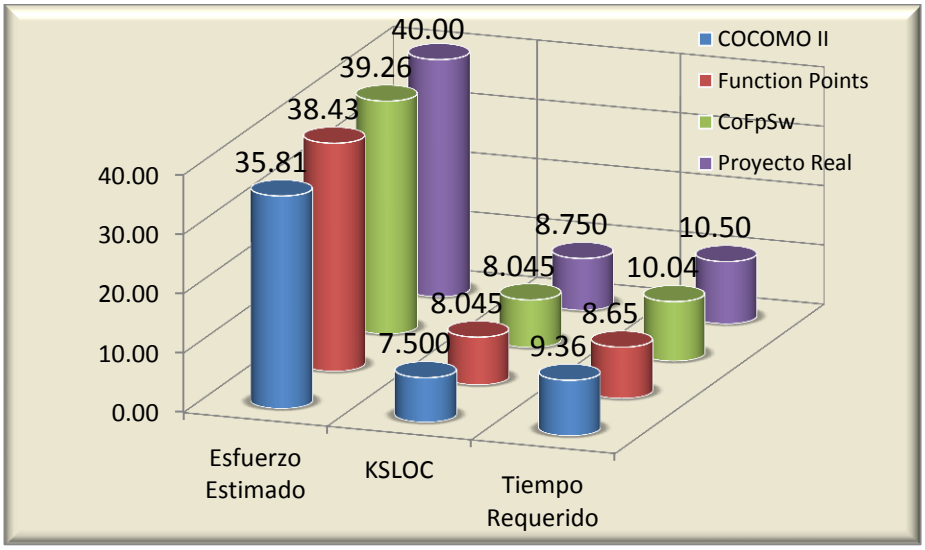


Figura 29. Gráfica de resultados comparación entre métricas.

Sin embargo es comprensible esperar que en pruebas de laboratorio, se obtengan resultados muy optimistas. Como puede apreciarse en la figura 30, ninguno de los tres métodos automatizados acertó en la estimación, pero sería correcto afirmar que el caso que más se acercó al valor real fue el que aplica la opción Cofpsw, implementada en la herramienta que plantea este documento.

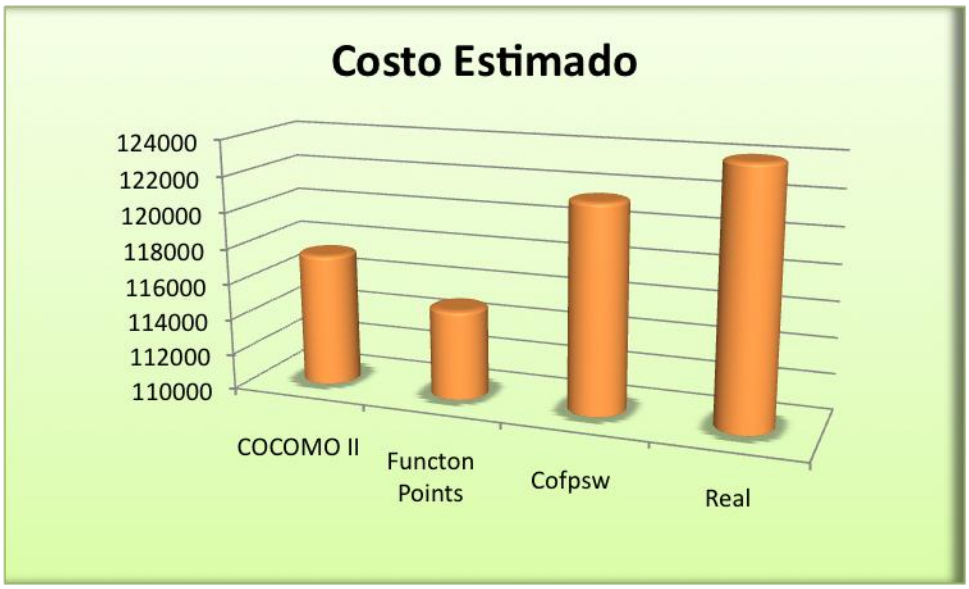


Figura 30. Gráfica comparativa de Costos Estimados

5.6 Resumen

Todas las pruebas de la aplicación fueron ejecutadas tomando como referente los que aparece en la literatura del estado del arte, y lo que publican los precursores de estas métricas. En teoría este trabajo demostró estar dentro del margen de resultados positivos para las estimaciones.

Con lo que se ha presentado en este documento, podría intentarse sugerir al director o ingeniero de desarrollo de un software, que entre otras cuestiones debería al menos tener claros los siguientes conceptos:

1. Existe un tiempo mínimo de desarrollo. Su valor descansa en un sistema de ecuaciones basado en unas constantes que permiten predecirlo para cada caso de forma fiable.
2. Intentar dirigir un proyecto más rápido de lo que su límite permite es imposible, o al menos, nadie lo ha conseguido.
3. Incluso intentar dirigir un proyecto para cumplir con el tiempo mínimo de desarrollo implica un esfuerzo sobrehumano, un sustancial incremento de los costes y un aumento de los defectos en la aplicación desarrollada.
4. Emplear elementos humanos en cualquier tarea y pretender ajustar su comportamiento a una “variable” del problema, conlleva su grado de Riesgo.
5. El empleo de variables susceptibles a manejarse estadísticamente, ofrece la posibilidad de tener una estimación aún más certera.

Capítulo VI
Conclusiones

6.1 Introducción

El Instituto, tradicionalmente ha afirmado que su misión principal es disponer para la sociedad, los logros tecnológicos a los que llega. Este tipo de documentos podrían ser considerados una especie de instructivos y las investigaciones que contienen, son esos logros que deben siempre estar al alcance de todos. Visto de esa manera, se encontraría entonces en este apartado, algo así como una breve reseña de lo que con ésta investigación se ha conseguido. Una explicación a manera de puntos de ventaja, lo que puede uno esperar del producto entregado.

En un sentido práctico, se enumeran en éste capítulo los productos o metas logradas, los fines que perseguían los autores al desarrollarlos, a manera de Conclusiones, así como la utilidad que podrían representar para un hipotético observador. Se plantean también los beneficios de adquirirlo en el apartado de Aportaciones. Finalmente podríamos estar anticipando a manera de anuncio, lo que éste hipotético observador podría esperar de la siguiente versión de tal producto, en el párrafo donde se expone el posible Trabajo Futuro.

6.2 Metas Logradas.

En la propuesta original se planteó, diseñar un prototipo de herramienta que fusionara dos de las métricas existentes para la evaluación de costos en la etapa de desarrollo de proyectos de software, que son COCOMO II e IFPUG. Dicha herramienta llamada CoFpSw (Cocomo II and Function Points Software), confeccionada mayormente en Visual Basic .Net, GUI de Microsoft en su versión 2010, ofrece la opción de aplicar en un proyecto de software, cualquiera de las tres metodologías (Las dos evaluadas y la propuesta por ésta tesis) indistintamente y que el usuario de dicha herramienta cuente con una estimación del costo de realizar tal proyecto

Se ha probado en diferentes foros relacionados con la industria del software, la utilidad de aplicar estas métricas en la planificación y elaboración de algún proyecto de software. Llevar al cabo la fusión en cierta medida de dos de las más importantes métricas, con la finalidad de facilitar más el trabajo a los usuarios de cualquiera de ellas, se considera un logro en el sentido de que la labor del investigador es precisamente coadyuvar a facilitar la labor humana en cualquier ámbito, es decir, poner la tecnología al servicio de la patria.

A principios del mismo mes en que se está terminando de escribir el presente documento, se publicó en la Revista Digital de la UNAM (Universidad Nacional Autónoma de México) (Vol 12, Número 6, ISSN 1067-6079), un artículo clasificado en Tecnologías de la Información y Comunicación, que con motivo de la presente investigación se sometió a revisión en meses pasados.

Dicho artículo se envió y se publicó con el Título: "*Prototipo de una herramienta de apoyo para la estimación de costos, en la etapa de desarrollo de un proyecto de software*". Pero además de éste, se sometieron a revisión dos más con los títulos: "*Prototipo de una herramienta que fusiona dos métricas de software empleadas para la estimación de costos.*" Y "*Merger of COCOMOII and IFPUG metrics in a software tool for Spanish-speaking users.*" Éste último devuelto para revisiones en la traducción y del otro se espera respuesta aún por parte del comité de arbitraje.

6.3 Conclusiones

Se analizaron las principales metodologías existentes en el mercado y se resaltó el uso de dos de ellas, que por su adaptabilidad y versatilidad son las más utilizadas. Después del análisis y profundización en sus procedimientos, pudo determinarse una posibilidad de mejorarlos, aplicando un método estadístico al manejo de los valores que utilizan para calcular sus estimaciones.

Se elaboró la herramienta software en una versión prototipo, que automatiza las principales funciones de dos de las métricas para la estimación de software más utilizadas. Esta herramienta ofrece una tercera opción de evaluación de los parámetros que utilizan tales metodologías y las combina con el fin de obtener mejores resultados.

Se comprobó mediante la utilización de la herramienta CoFpSw, la posibilidad de aportar una significativa mejora en los procesos de planificación y desarrollo de un proyecto de software. Toda vez que los resultados emitidos por dicha aplicación, se mantuvieron dentro de un margen de aceptación.

6.4 Aportaciones.

Al proponer una herramienta de software de índole administrativa, debido al nivel con el que puede involucrarse en la fase de planeación, se plantea la opción de provocar que la manera de hacer software en México se ajuste más a los lineamientos y objetivos de la organización, cuando ésta pertenece al giro de la industria del Software.

Las políticas y procedimientos de la organización, pueden ser tan puntuales y al mismo tiempo tan flexibles, que rijan y controlen los tiempos y movimientos de quienes participan en ella, sin intervenir en su autonomía o restringir sus libertades. Del mismo modo, sujetar cualquier actividad a este nivel de control, puede ser benéfico y se reflejará en los resultados; al ser positivo para la tarea en sí, para quien se sirve de ella y para la organización entera.

Dicho de otra forma y puntualizando, puede observarse qué a raíz de la presente investigación:

- Se sentaron precedentes para utilizar lo que podría considerarse una nueva metodología de evaluación de proyectos de software.
- Se cuenta con una herramienta de software que facilita el empleo de al menos dos de las métricas más utilizadas en la estimación de costos de desarrollos de software
- Con éste producto, podría estarse provocando que ingenieros de software, desarrolladores y programadores que nunca han aplicado alguna metodología en la planeación de sus proyectos, ahora lo hagan, al utilizar la herramienta.

6.5 Trabajo Futuro.

Para facilitar aún más cualquier tipo de actividad, se busca automatizarla y de ser posible hacer que la sistematización de dicha actividad involucre cada vez más una toma de decisiones automática. Ante esta idea como objetivo, podría sugerirse involucrar algunos aspectos relacionados con la inteligencia artificial en el presente proyecto, para lograr que la herramienta sea totalmente interactiva y obedezca a dar respuestas y ofrecer soluciones, ante comandos de voz.

El objetivo primordial de la Ingeniería de Software es producir un sistema, aplicación o producto de alta calidad. Para lograr este objetivo, los ingenieros de software deben emplear métodos efectivos junto con herramientas modernas dentro del contexto de un proceso maduro de desarrollo del software. Así, herramientas de este tipo harán que un buen ingeniero y administrador del software sea capaz de producir y entregar productos de alta calidad, en el tiempo acordado, que cubran los requerimientos del usuario y que ello le reporte utilidades apropiadas.

Como puntos específicos en cuanto a la herramienta podríamos señalar:

- Terminar el módulo de “Archivos de Ayuda” para todo el sistema.
- Que la herramienta muestre los pasos de cómo aplica la Estadística.
- Agregarle un módulo de reconocimiento de instrucciones dictadas por medio de un micrófono
- Que eventualmente sea capaz de integrar rutinas de evaluación que siguiera cualquier otra metodología que demuestre ser tan eficiente o incluso mejor que las abordadas en éste documento.

❖ Bibliografía:

[1] Ian Sommerville,
Software Engineering,
8ª Edición. ISBN: 13 978-0321313799
Addison Wesley, 2008

[2] William A. Florac and Anita D. Carleton,
Measuring the software process: "statistical process control for software process improvement ",
ISBN: 0–201–60444–2.
Addison Wesley, (1999).

[3] Barry W. Boehm and Kevin J. Sullivan,
Software Engineering Economics,
1ª Edición: ISBN: 0138221227
Prentice Hall, 1981

[4] Stephen H. Kan.
Metrics and models in software quality engineering,
2ª Edición: ISBN: 0–201–72915–6.
Adisson wesley, 95–456 (2002).

[5] Roger S. Pressman,
Ingeniería de Software "Un enfoque práctico"
4ª Edición. ISBN: 0-07-709677-0
McGrawHill, (1998)

[6] Garmus and David Herron.
Measuring the software process. "A practical guide to functional measurements",
3ª Edición. ISBN: 0–201–69944–3.
Addison wesley, 2001.

[7] Barry W. Boehm, Clark Horowitz Brown,
Software Cost Estimation with Cocomo II
Prentice Hall NJ, USA 2000

[8] Fairley Richard E.
Software Engineering Concepts.
McGraw Hill/Interamericana de México, (1988)

[9] Jones, Capers.
Applied Software Measurement: Global Analysis of Productivity and Quality
3ª Edición. . ISBN: 978-0-07-150244-3
McGraw-Hill, USA, 1986

❖ Referencias:

[10] Albrecht, A. J. *Measuring Application Development Productivity,* IBM Applications Development Symposium, Monterey, CA, USA, 1979.

[11] Symons, Charles R. *Function Point Analysis: Difficulties and Improvements* IEEE Transactions on Software Engineering, vol. 14, No.1, January 1988

[12] Software Engineering Standards Committee. *IEEE Standard for Software Maintenance,* IEEE-SA Standards Board June 1998

[13] Boehm, B., *COCOMO II Model Definition Manual,* Software Engineering Economics. 1981.

[14] Albrecht, A. J. and Gaftney, J. E., *Software Function, Source Lines of Code, and Development Effort Prediction: A SoftwareScience Validation*, IEEE Transactions on Software Engineering, SE-9(2), 1983,

[15] Boehm, B., *An Overview of the COCOMO 2.0 Software Cost Model*, 1999

[16] J. Kaczmarek, M. Kucharski, *Size and effort estimation for applications written in* Java, Information and Software Technology, SE-46 (589–601), 2004

[17] IFPUG *Counting Practices Manual, Release 4*, International Function Point Users Group, Westerville, OH; 1995

[18] Kazman, R., Klein, M., Barbacci, et al., *The Architecture Tradeoff Analyisis Method*., IEEE, ICECCS, 1998

[19] L. De Rore, et al., *Cocomo II as productivity measurement: a case study at KBC,* Katholieke Universiteit Leuven, Bel, 2003

[20] Symons, C. R., *Function-point analysis: difficulties and improvements,* IEEE Trans. on Software Engineering, 14(1), 1992

[21] Fenton, N.E., *Software Metrics u Rigorous Approach*, Chapman & Hall, London.

[22] Humphrey, W.S., *Characterizing the software process: a maturity framework*. IEEE Software 5(2), 1992

- Referencias Web:

[23] http://www.ifpug.org/educational/

[24] http://www.qsm.com/tools/slim-estimate

[25] http://www.charismatek.com/_public4/html/services/service_fpa.htm

[26] http://www.totalmetrics.com/function-points/Scope-Counting-59.swf

[27] http://www.totalmetrics.com/consulting-function-points/project-cost-estimation

[28] http://www.aemes.org/index.php

[29] http://www.aemes.org/index.php/inscripciones

[30] http://www.aemes.org/index.php/revista-de-procesos-y-metricas/

[31] http://www.ibm.com/developerworks/rational/library/edge/08/jan08/tarbet/#N100D4

[32] http://www.reifer.com/cocomo_eff.htm

[33] http://www.softstarsystems.com

[34] http://www.softwaremetrics.com/services.html

[35] http://sunset.usc.edu/COCOMOII/cocomo81_pgm/cocomo81.html,

[36] http://sunset.usc.edu/COCOMOII/cocomo.html

[37] http://msdn.microsoft.com/en-us/library/bb421527.aspx

[38] http://yunus.hun.edu.tr/~sencer/cocomo.html

[39] http://www.stsc.hill.af.mil/crosstalk/1995/jun/metrics.asp

[40] http://www.webcomtechnology.com.mx/FabricaSw.html

[41] http://www.galorath.com/index_es.html

[42] http://www.galorath.com/flash_presentations/sem_web_demo/

[43] http://ingenieriasoftware1.blogspot.es

[44] http://zaurak.cis.ksu.edu/~radhika/hw2.html

[45] http://www.sap.com/mexico/solutions/index.epx

Referencias actualizadas al día 22 de julio de 2011

❖ Glosario

API	(Application Program Interface). Conjunto de convenciones internacionales que definen cómo debe invocarse una determinada función de un programa desde una aplicación.
ArgoUML	Aplicación de diagramado de UML escrita en Java y publicada bajo la Licencia BSD.
ASMA	Australian Software Metrics Association
COCOMO	(Constructive Cost Model) Modelo de estimación de costos propuesto por Boehm
CARE	Computer Aided Requirements Engineering
CASE	Computer Aided Software Engineering
Corel Draw	Aplicación informática de manejo vectorial flexible, relativamente de la corporación Corel, diseñada para el dibujo, la maquetación y/o la publicación.
COSMIC	(Common Software Metrics International Consortium)
Framework	Estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado.
IFPUG	International Function Point Users Group
HTML	(HyperText Markup Language) Lenguaje de Marcas de Hipertexto.
Métrica.	Evaluación del grado en el cual un producto o proceso posee un atributo determinado extensión, cantidad, dimensiones, capacidad o tamaño) (IEEE, 1993).
Medición.	Proceso objetivo y empírico por el que se asignan números o símbolos a atributos de entidades del mundo real con objeto de describirlas (Fenton y Kitchenham, 1991).
MsWord	Aplicación de procesamiento de texto desarrollado por Microsoft.
MySql	Software libre: Sistema de gestión de base de datos relacional, multihilo y multiusuario.
NESMA	Netherlands Software Metrics Users Association
OASIS	Open and Active Specification of Information Systems
ODBC	Open Data Base Connection
RUP	Rational Unifed Process
Plug-in	(Enchufable) Aplicación de software que es un cComplemento de una aplicación.
SLOC	source lines of code
Socket	Concepto abstracto de un "objeto" por el cual dos programas (Software) (posiblemente situados en computadoras distintas) pueden intercambiar cualquier flujo de datos
UML	Unified Modeling Language
UPV	Universidad Politécnica de Valencia
WebML	Web Modeling Language

❖ Índice de Acrónimos

A/S	Algorithmic/Scientific
ACAP	Capacidad de los analistas, para evaluar las variables de costo
APEX	Experiencia en aplicaciones similares.
ASI	Análisis del Sistema de Información
BFC	Base Functional Component
CBO	Coupling Between Objects
CCPMi	Cantidad de caracteres promedio del modelo de entrada i (caracteres/modelo)
CCPRk	Cantidad de caracteres promedio del registro k (caracteres/modelo)
CFPS	Certified Function Point Specialist
CHk(x)	Cantidad de hombres necesarios para la etapa k del sistema x
CLPMj	Cantidad de líneas promedio del modelo de salida j (líneas/modelo)
CMMk	Cantidad máxima de modelos del registro k a archivar
$CONS_{i,J}$,K	Consumo del Producto j en la actividad k en el per. i
COP's	Component object points
CPLX	Complejidad del software planeado
CSI	Construcción del Sistema de Información
CTE_{i-1}	Cantidad de trabajadores que egresarán en el mes i
CT_{i-1}	Cant. trabajadores mes anterior al mes i
CTI_{i-1}	Cantidad de trabajadores que ingresan en el mes i
CV	Convergent Validity
DATA	Magnitud del manejo y/o acceso a datos
DESI	Desarrollo de Sistemas de Informcaión
DIT	Depth of Inheritance Tree
DOCU	Cantidad de documentación a realizar
DSI	Diseño del Sistema de Información.
DV	Discriminant Validity
DWPs	Data Web Points
EI	External Input
EIF	External Interface File
EO	External Output
EQ	External Inquiry
ESE	Empirical Software Engineering
ESF	Esfuerzo (Horas / Hombre)
ESFk(x)	Esfuerzo proyección de la etapa k del Sistema x
EVS	Diseño del Sistema de Información
FEi	Frecuencia de entrada del modelo i (Modelos/mes)
FFP	Full Function Points
FLEX	Flexibilidad del Desarrollo (Variable de Esfuerzo)
FPA	Function Point Analysis
FPs	Function Points

FSM	Functional Size Measurement
FSMj	Frecuencia de salida del modelo j (modelos/mes)
FTR	File Types Referenced
FURs	Functional User Requirements
GQM	Goal/Question/Metric
IAS	Implantación y Aceptación del Sistema
ID	Interaction Diagram
IDE	Integrated Development Environment
IEC	International Electrotechnical Commission
IF	Identification Function
IIU	Instance Interaction Unit
ILF	Internal Logical File
ISO	International Organization for Standardization
ITU	Intention To Use
LTEX	Experiencia específica en ese lenguaje y herramientas
m	Cantidad de meses del período a calcular.
MAM	Method Adoption Method
MDIU	Master-Detail Interaction Unit
MEM	Method Evaluation Method
MIS	Management Information Systems
MRE	Magnitude of Relative Error
MSI	Mantenimiento del Sistema de Información
n	Cantidad de Períodos registrados.
NC_j,k :	Norma de Consumo del Producto j en la actividad k
OMT	Object Modeling Technique
OO	Object-Oriented
OOH	Object-Oriented Hypermedia
OOHDM	Object-Oriented Hypermedia Design Method
OOmFP	OO-Method Function Points
OOmFPWeb	OO-Method Function Points for the Web
OOWS	Object-Oriented Web Solutions
PCAP	Capacidad para programar ese tipo de aplicaciones
PCON	Frecuecia del Rol del personal
PEOU	Perceived Ease of Use
PIU	Population Interaction Unit
PLEX	Experiencia en la plataforma elegida
PMS	Project Management System
POP	Predictive Object Points
PPI	Plan Perspectivo Informático
PSI	Planificación de Sistemas de Información
PU	Perceived Usefulness
PREC	Precedencia (Variable de Esfuerzo)
PVOL	Frecuencia en cambios en la plataforma empleada
RE	Rapidez de Entrada de Datos (Velocidad) (caracteres/seg.)

RELY	Plan emergente de costo por fallas
REP	REProducibility measurements
RET	Record Element Type
RMC	(Rational Method Composer)
RS	Rapidez de Salida de Datos (líneas/min.)
RT	Real-time embedded
RUR	Reference User Requirements
RUSE	Reusabilidad probable del software
SCED	Nivel de planificación del desarrollo
SE	Software Engineering
SITE	Nivel de integración – comunicación o de situación de un equipo "none in Site"
STD	State Transition Diagram
STOR	Requerimientos de memoria
TAM	Technology Acceptance Model
TC	Tiempo computacional (Tiempo que trabaja la Unidad Central) (horas/mes).
TDES	Tiempo del desarrollo (en meses)
TDESk(x)	Tiempo de Desarrollo de la etapa k del sistema x
TEA	Tiempo de Ejecución y Acceso (horas/mes).
TED	Tiempo de entrada de datos (horas/mes)
TIME	Restricciones de tiempo de proceso.
TOOL	Grado de sofisticación de las herramientas a utilizar.
TRA	Theory of Reasoned Action
TSD	Tiempo de salida de datos (horas/mes)
UFP	Unadjusted Functions Points
UKSMA	United Kingdom Metrics Association
VIE	Volumen de información de entrada (caracteres/mes)
VIR	Volumen de información de los registros (caracteres)
VIS	Volumen de información de salida (caracteres/mes)

❖ Anexos

A. Script de la Base de Datos

```
-- MySQL Administrator dump 1.4
-- ------------------------------------------------------
-- Server version   5.2
-- ------------------------------------------------------
-- Author: Antonio Ramírez
-- ------------------------------------------------------
-- Inicia Directivas de Servidor SQL

/* Creación de la Base de Información */;
--
-- Creando schema para COFPSW
--

CREATE DATABASE IF NOT EXISTS cofpswdb;
USE cofpswdb;
--
-- Definición de la tabla formulas
--
DROP TABLE IF EXISTS Formulas;

CREATE TABLE Formulas (
  id_Form int(11) NOT NULL,
  nom_Form varchar(80) DEFAULT NULL,
  PRIMARY KEY (id_Form)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
--
-- Relleno de datos de la tabla Fórmulas
--
/*!40000 ALTER TABLE Formulas DISABLE KEYS */;
INSERT INTO Formulas (id_Form, nom_Form) VALUES
 ( 1, 'Cantidad de hombres necesarios' ),
 ( 2, 'Norma de Consumo' ),
 ( 3, 'Plan de Abastecimiento Técnico de Material' ),
 ( 4, 'Presupuesto de la Inversión' ),
 ( 5, 'Presupuesto de Salarios' );
/*!40000 ALTER TABLE Formulas ENABLE KEYS */;
--
-- Definición de la tabla VariablesCO
--
DROP TABLE IF EXISTS VariablesCO;
CREATE TABLE VariablesCo (
  id_Var int(10) DEFAULT NULL,
  id_Proy int(10) DEFAULT NULL,
  id_User int(10) DEFAULT NULL,
  val_Acap int(10) DEFAULT NULL,
  val_Apex int(10) DEFAULT NULL,
  val_Pcap int(10) DEFAULT NULL,
  val_Plex int(10) DEFAULT NULL,
  val_Ltex int(10) DEFAULT NULL,
  val_Pcon int(10) DEFAULT NULL,
```

```
  val_Tool int(10) DEFAULT NULL,
  val_Site int(10) DEFAULT NULL,
  val_Sced int(10) DEFAULT NULL,
  val_Time int(10) DEFAULT NULL,
  val_Stor int(10) DEFAULT NULL,
  val_Pvol int(10) DEFAULT NULL,
  val_Rely int(10) DEFAULT NULL,
  val_Data int(10) DEFAULT NULL,
  val_Cplx int(10) DEFAULT NULL,
  val_Ruse int(10) DEFAULT NULL,
  val_Docu int(10) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
--
-- Relleno de datos de la tabla VariablesCO
--
/*!40000 ALTER TABLE VariablesCO DISABLE KEYS */;
INSERT INTO VariablesCO (id_Var,id_Proy,id_User,val_Acap,val_Apex,
val_Pcap,val_Plex,val_Ltex,val_Pcon,val_Tool,val_Site,val_Sced,
val_Time,val_Stor,val_Pvol,val_Rely,val_Data,val_Cplx,val_Ruse,
val_Docu) VALUES
 (1,2,1,1,2,3,3,5,1,2,1,1,2,3,3,5,2,1,1,2),
 (1,2,3,5,1,2,1,1,2,3,3,5,2,1,1,1,1,2,3,3);
/*!40000 ALTER TABLE VariablesCO ENABLE KEYS */;
--
-- Definición de la tabla VarCoNom
--
DROP TABLE IF EXISTS VarCoNom;
CREATE TABLE VarCoNom (
  nom_Acap varchar(80) DEFAULT NULL,
  nom_Apex varchar(80) DEFAULT NULL,
  nom_Pcap varchar(80) DEFAULT NULL,
  nom_Plex varchar(80) DEFAULT NULL,
  nom_Ltex varchar(80) DEFAULT NULL,
  nom_Pcon varchar(80) DEFAULT NULL,
  nom_Tool varchar(80) DEFAULT NULL,
  nom_Site varchar(80) DEFAULT NULL,
  nom_Sced varchar(80) DEFAULT NULL,
  nom_Time varchar(80) DEFAULT NULL,
  nom_Stor varchar(80) DEFAULT NULL,
  nom_Pvol varchar(80) DEFAULT NULL,
  nom_Rely varchar(80) DEFAULT NULL,
  nom_Data varchar(80) DEFAULT NULL,
  nom_Cplx varchar(80) DEFAULT NULL,
  nom_Ruse varchar(80) DEFAULT NULL,
  nom_Docu varchar(80) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
--
-- Relleno de datos de la tabla VarCoNom
--
/*!40000 ALTER TABLE VarCoNom DISABLE KEYS */;
INSERT INTO VarCoNom (nom_Acap, nom_Apex, nom_Pcap,
  nom_Plex, nom_Ltex, nom_Pcon, nom_Tool, nom_Site,
  nom_Sced, nom_Time,     nom_Stor, nom_Pvol, nom_Rely,
  nom_Data, nom_Cplx,     nom_Ruse, nom_Docu) VALUES
 ( 'Capacidad de los analistas, para evaluar las variables de costo.',
   'Experiencia en aplicaciones similares.',
```

```
   'Capacidad para programar ese tipo de aplicaciones.',
   'Experiencia en la plataforma elegida.',
   'Experiencia específica en ese lenguaje y herramientas.',
   'Frecuecia del Rol del personal.',
   'Grado de sofisticación de las herramientas a utilizar.',
   'Nivel de integración – comunicación de equipo "none in Site".',
   'Nivel de planificación del desarrollo.',
   'Restricciones de tiempo de proceso.',
   'Requerimientos de memoria.',
   'Frecuencia en cambios en la plataforma empleada.',
   'Plan emergente de costo por fallas.',
   'Magnitud del manejo y/o acceso a datos.',
   'Complejidad del software planeado.',
   'Reusabilidad probable del software.',
   'Cantidad de documentación a realizar.');
/*!40000 ALTER TABLE VarCoNom ENABLE KEYS */;
--
-- Definición de la tabla VariablesPF
--
DROP TABLE IF EXISTS VariablesPF;
CREATE TABLE VariablesPF (
  id_Var int(11) DEFAULT NULL,
  nom_Var varchar(40) DEFAULT NULL,
  desc_Var varchar(40) DEFAULT NULL,
  val_Var varchar(10) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
--
-- Relleno de datos de la tabla VariablesPF
--
/*!40000 ALTER TABLE VariablesPF DISABLE KEYS */;
INSERT INTO VariablesPF (id_Var, nom_Var, desc_Var, val_Var) VALUES
 ( 1, 'temporal name','desc?', 'Muy Bueno' );
/*!40000 ALTER TABLE VariablesCO ENABLE KEYS */;
--
-- Definición de la tabla Personal
--
DROP TABLE IF EXISTS Personal;
CREATE TABLE Personal (
  id_User int(11) NOT NULL,
  num_User varchar(4) DEFAULT NULL,
  nom_User varchar(40) DEFAULT NULL,
  nivel_User int(10) unsigned DEFAULT NULL,
  PRIMARY KEY (id_User)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
--
-- Llenado de Datos de la tabla Personal
--
/*!40000 ALTER TABLE Personal DISABLE KEYS */;
INSERT INTO Personal (id_User,num_User,nom_User,nivel_User) VALUES
 (1, 01, 'Invitado', 1),
 (2, 02, 'Antonio Ramírez Ramírez', 3),
 (3, 03, 'Sandra Morales Güitrón', 3),
 (4, 04, 'tonotron', 3),
 (5, 05, 'sandra', 3),
 (6, 06, 'cesar', 3);
/*!40000 ALTER TABLE Personal ENABLE KEYS */;
```

```
--
-- Definición de la tabla Proyecto
--
DROP TABLE IF EXISTS Proyecto;
CREATE TABLE Proyecto (
  id_Proy int(11) NOT NULL,
  num_Proy varchar(4) DEFAULT NULL,
  nom_Proy varchar(40) DEFAULT NULL,
  tam_Proy int(10) unsigned DEFAULT NULL,
  met_Proy varchar(40) DEFAULT NULL,
  PRIMARY KEY (id_Proy)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
--
-- Llenado de Datos de la tabla Proyecto
--
/*!40000 ALTER TABLE Proyecto DISABLE KEYS */;
INSERT INTO Proyecto (id_Proy, num_Proy, nom_Proy, tam_Proy, met_Proy) VALUES
 (1, 01, 'Primero' , 2000, 'COFP' ),
 (2, 02, 'Segundo' , 3000, 'COFP'),
 (3, 03, 'Tercero' , 3500, 'COFP'),
 (4, 04, 'Cuarto'  , 3800, 'COFP'),
 (5, 05, 'Quinto'  , 4300, 'COFP');
/*!40000 ALTER TABLE Proyecto ENABLE KEYS */;

--
-- Definición de la tabla VALORES
--
DROP TABLE IF EXISTS valores;
CREATE TABLE valores (
  id_Var int(10) NOT NULL,
  Absoluto int(10) DEFAULT '100',
  MuyAlto int(10) DEFAULT '80',
  Alto int(10) DEFAULT '70',
  Medio int(10) DEFAULT '50',
  Bajo int(10) DEFAULT '35',
  MuyBajo int(10) DEFAULT '25',
  Nulo int(10) DEFAULT '0',
  PRIMARY KEY (id_Var)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
--
-- Llenado de Datos de la tabla ACAP
--
/*!40000 ALTER TABLE VALORES DISABLE KEYS */;

INSERT INTO valores (id_Var, Absoluto, MuyAlto, Alto, Medio, Bajo, MuyBajo, Nulo)
VALUES
 (1,'100', '80', '70', '50', '35', '25', '0');
/*!40000 ALTER TABLE VALORES ENABLE KEYS */;
select * from VarCoNom;
select * from valores;
```

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México, el día 26 de Julio de 20011, el que suscribe, Lic. Antonio Ramírez Ramírez, alumno del Programa de Maestría en Ciencias de la Computación, con número de registro B091653, adscrito al Centro de Investigación en Computación, manifiesta que es único autor intelectual y material del presente trabajo de Tesis bajo la dirección de la Maestra en Ciencias Sandra Dinora Orantes Jiménez y que en esta acto cede todos los derechos del trabajo intitulado "*Prototipo de una herramienta de apoyo para la estimación de costos de un proyecto de software*" al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo, sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección tonotron@gmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Lic. Antonio Ramírez Ramírez