ESCUELA POLITÉCNICA DEL EJÉRCITO

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

"Propuesta Metodológica de la Estimación por Conteo de Puntos de Función basados en el Modelo Entidad-Relación".

Previa a la obtención del Título de:

INGENIERO EN SISTEMAS E INFORMÁTICA

POR: ROSA TATIANA VALENZUELA VARELA

SANGOLQUÍ, 26 de Septiembre del 2008

CERTIFICACIÓN

Certifico que el presente trabajo fue realizado en su totalidad por la Srta. Valenzuela Varela Rosa Tatiana como requerimiento parcial a la obtención del título de INGENIERO EN SISTEMAS E INFORMÁTICA.

Sangolquí, 26 de Septiembre del 2008

_____

Ing. Geovanny Raura
Director de Tesis

AGRADECIMIENTO

Agradezco a mi madre que me ha permitido cursar mi carrera estudiantil y universitaria, apoyándome siempre en cada decisión que he tenido en mi vida; a mis hermanos que lejos o cerca han sabido darme su apoyo sincero y han seguido paso a paso mi caminar; y a mi abuelita que con sus bendiciones me ha protegido siempre.

Muchas gracias a mis maestros por todo el tiempo dedicado a la docencia, y por haberme impartido valiosos conocimientos en las aulas de clase.

Finalmente agradezco la presencia de compañeros y amigos con quienes he vivido experiencias inolvidables.

ROSA TATIANA VALENZUELA VARELA

DEDICATORIA

El siguiente proyecto va dedicado a mi familia, especialmente a mi madre porque es el pilar fundamental en mi vida; gracias a ella soy la mujer con valores y principios bien cimentados; a mis hermanos de quienes he tenido siempre su apoyo incondicional y respaldo, a mi abuelita que en silencio reza y pide por cada uno de sus hijos.

Les dedico también a las personas que siguieron de cerca el desarrollo de este proyecto, a mis amigos y compañeros, ya que supieron brindarme su amistad fraterna y leal.

ROSA TATIANA VALENZUELA VARELA

ÍNDICES

ÍNDICE DE CONTENIDOS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

RESUMEN

En el complejo mundo del desarrollo de software empresarial se resalta entre otras, como principal métrica para medir el tamaño de un sistema el indicador, Punto de Función (PF) definido por Allan Albrecht a finales de la década del setenta. No cabe duda que la información entregada por el citado indicador es importante al momento de estimar aspectos funcionales en el desarrollo de productos de Software; sin embargo esta técnica no está asociada a los aspectos complementarios a la funcionalidad, como por ejemplo la definición de una Arquitectura de Software que asegure el cumplimiento de patrones de diseño, escalabilidad, altos niveles de reusabilidad entre otras.

En el presente proyecto se presenta un método alternativo denominado Model Prototype Estimation (MPE), con el objetivo principal de determinar el esfuerzo requerido para desarrollar arquitecturas empresariales de software; el método es un híbrido de estimación, que comienza por el conteo de Puntos de Función aplicado a un Modelo Conceptual y considerando los cuatro métodos básicos como elementos de función (adicionar, modificar, eliminar y consultar); luego se define una Arquitectura base y se genera un Prototipo funcional que permite calcular el número total de líneas de código. Finalmente se estima el esfuerzo, tiempo y personal mediante la aplicación del modelo COCOMO propuesto por Barry Boehm.

1 CAPÍTULO I - INTRODUCCIÓN

El desarrollo de aplicaciones bajo el paradigma orientado a objetos propuesto por Booch (1994), Jacobson (1992), Rumbaugh (1991) y otros, se orientan hacia el logro de requerimientos funcionales, sin prestar mayor atención a los requerimientos no funcionales o atributos de calidad, por lo que ha emergido una disciplina relativamente nueva en la ingeniería de software denominada arquitectura de software (Losavio 2006).

El desarrollo del software moderno se basa cada vez más en un enfoque centrado en la arquitectura, existiendo propuestas como RUP[1], Bosch[2] y ATAM[3], que incluyen como factor determinante en la toma de decisiones a la calidad en el producto influenciados por la arquitectura.

De acuerdo a la IEEE Std 1471-2000, se define una arquitectura de software como:

"La organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implementarán, y los principios que orientan su diseño y evolución"[4].

Es decir la arquitectura aporta una visión abstracta del sistema de alto nivel, donde el detalle de cada uno de los módulos definidos, se pospone para etapas posteriores de diseño.

Cuando el tamaño de los sistemas se incrementa, los algoritmos y las estructuras de datos no representan el mayor problema, siendo la arquitectura la que encierra el mayor reto para los ingenieros de software. Entre los temas a resolver están los protocolos de comunicación, sincronización y acceso a datos; la asignación de responsabilidades a los elementos de diseño, distribución física; la escalabilidad y el rendimiento; por mencionar algunos.

[1] (Rational Unified Process, es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos - http://es.wikipedia.org/wiki/RUP)
[2] (Bosch 2000, Szyperski 1998)
[3] (Attribute Trade off Analysis Method - Kazman, Klein, Barbacci, Longstaff, Lipson y Carriere, 1998)
[4] (IEEE Recommended Practice for Architectural Description of Software-Intensive Systems, 2000)

La determinación del tamaño de los sistemas, es uno de los factores claves dentro de las actividades de la gestión de proyectos; en principio el empleo de la medida del número de líneas de código SLOC[1], ha sido la métrica tradicional para estimar el esfuerzo y tiempo de desarrollo, medida criticada básicamente por su dependencia del lenguaje, lo que dificulta su uso en etapas tempranas de desarrollo y finalmente por la no existencia de una definición aceptada de manera general.

Para solventar este inconveniente, Allan Albrecht, propone la métrica de Punto de Función (PF), en procura de obtener una estimación del tamaño del software *"independiente de la tecnología utilizada para su desarrollo y dependiente, únicamente de la funcionalidad del sistema"*[2]. Lo que hace suponer que la arquitectura es relegada a un segundo plano.

Se define entonces un método denominado MPE[3], que es un híbrido entre el uso de modelos conceptuales, la generación de un prototipo, y la estimación por puntos de función y el cálculo de líneas de código. El método es aplicado a diferentes casos de estudio, y los resultados obtenidos bajo este enfoque se presentan como base para futuros estudios de validación del método.

1.1 Antecedentes

El desarrollo del software requiere de la estimación como una herramienta para controlar y administrar los recursos que se necesitan y que se utilizan antes y durante el proyecto.

No se puede considerar a la estimación como una ciencia exacta ya que existen numerosas variables humanas, técnicas, del entorno y políticas, entre otras, que intervienen en su proceso y que pueden afectar los resultados finales. Sin embargo, cuando es llevada a cabo en forma sistemática, se pueden lograr resultados con un grado aceptable de riesgo y convertirla en un instrumento útil para la toma de decisiones.

[1] (Source Lines Of Code)
[2] (Allan J. Albrecht and John E. Gaffney, November 1983. "Software Function, Lines of Code, and Development Effort Prediction: A Software Science Validation", IEEE Transactions on Software Engineering, vol SE-9, No 6)
[3] (Model Prototype Estimation)

Actualmente en el mercado tecnológico existen métodos y herramientas que permiten la estimación basados en puntos de función, teniendo como fundamento todos los requerimientos funcionales y no funcionales definidos en el documento de especificación de requerimientos.

Las tendencias informáticas actuales exigen la flexibilidad, interoperabilidad y rapidez en las aplicaciones, por lo que es necesario emplear metodologías, herramientas y técnicas de reusabilidad; aspectos no considerados en las metodologías y herramientas de estimación existentes.

Dentro del desarrollo del software, difícilmente se puede realizar una estimación certera de la cantidad de líneas de código que tendrá la aplicación, ya que dependen fuertemente del programador y la herramienta de desarrollo a utilizar.

Adicionalmente los puntos de función pueden ser estimados con mayor certeza a partir de una buena especificación de requisitos. Lamentablemente en el Ecuador la especificación de requerimientos no se la realiza de una manera apropiada, lo que trae como consecuencia una estimación no apegada a la realidad y por consiguiente ocasiona que muchos proyectos software inicien, pero que jamás concluyan. Cuando los proyectos se plantean sobre la base de especificaciones vagas o inexistentes, es muy poco lo que se puede desarrollar en el futuro.

Por esta razón, se plantea un método diferente de estimación de un proyecto, tomando el Modelo Entidad-Relación como fundamento para el cálculo de los puntos de función básicos, luego se define una Arquitectura base y se genera un Prototipo funcional que permite calcular el número total de líneas de código. Finalmente se estima el esfuerzo, tiempo y personal mediante la aplicación del modelo COCOMO.

1.2 Justificación

La nueva metodología de estimación del software a desarrollar, presume una planificación de proyecto mucho más certera. La utilización del Modelo Entidad-Relación como base para el cálculo de los puntos de función, sirve de guía para

determinar la cantidad de entradas, salidas externas, consultas, archivos lógicos e interfaces que utilizará el proyecto de software, y buscará un equilibrio en la estimación de esfuerzos empleados tanto en la arquitectura como en la funcionalidad del sistema; siendo un método diferente en el cálculo de estimación usual basada solo en la funcionalidad.

La realidad actual de las empresas de software en nuestro país es crítica ya que no existe una técnica de estimación que se adapte a la necesidad de los proyectos de software; esto conlleva a que se realice la planificación y estimación de proyectos en base a la experiencia profesional del equipo de trabajo para definir esfuerzos y tiempos empleados (ver anexo A). Por esta razón, el análisis de la información obtenida, permitirá plantear los lineamientos que regirán la propuesta metodológica, enmarcada en lograr un proceso caracterizado con la simplicidad y fluidez en el cálculo de los puntos de función, que se ajuste a la realidad del país.

El planteamiento de este nuevo método exigirá una investigación de proyectos de desarrollo de software ejecutados en el Ecuador; constituyéndose como fuente principal los Modelos Entidad-Relación para la ejecución de la nueva propuesta.

1.3 Objetivos

1.3.1 Objetivo General

- Realizar una nueva propuesta metodológica, mediante el conteo de puntos de función, basados en el Modelo Entidad-Relación para mejorar el proceso de estimación del esfuerzo funcional y de arquitectura de los proyectos de software.

1.3.2 Objetivos Específicos

- Realizar la investigación de campo de las empresas desarrolladoras de software en el Ecuador, para conocer la situación actual sobre la estimación y planificación de proyectos.

- Generar los lineamientos de la propuesta metodológica para la estimación del Software, basado en el Modelo Entidad-Relación.

- Realizar los cálculos de puntos de función y estimación del esfuerzo funcional y de arquitectura de proyectos de software, mediante la nueva propuesta metodológica.

- Interpretar, analizar y comparar los resultados del esfuerzo funcional y de arquitectura obtenidos con la nueva propuesta metodológica, y los obtenidos con la metodología habitual de estimación funcional de proyectos.

- Utilizar el prototipo de estimación MPE, para optimizar el tiempo y recursos en los cálculos de la estimación de proyectos.

1.4 Hipótesis

La propuesta Metodológica de la estimación por Conteo de Puntos de Función basados en el Modelo Entidad-Relación permitirá determinar una estimación más precisa y real de los proyectos software; ya que considera importante el esfuerzo requerido en la arquitectura del sistema, que en ocasiones puede ser mayor al esfuerzo necesario para el desarrollo de los requisitos funcionales.

1.5 Alcance

El presente proyecto tiene como alcance la comprobación de la hipótesis planteada, para lo cual se debe realizar varias pruebas aplicando las metodologías tradicionales y la propuesta metodológica desarrollada; y realizar una comparación entre ellos.

Esta propuesta metodológica será aplicable únicamente a proyectos de software que interactúen con una base de datos; se requiere de estos proyectos al menos el modelo de datos conceptual, que será una fuente de información fundamental para proceder a los cálculos de estimación necesarios.

1.6 Concepto de Metodología

Como se ve a continuación, tanto en los objetivos que nos planteamos en esta tesis como en la hipótesis que pretendemos demostrar, se hace uso de la palabra "metodología", término cuyo significado tal vez resulte demasiado amplio y demasiado vago. Por esta razón, algunos autores la definen así: *"Una metodología debe disponer del soporte necesario para cubrir el ciclo de vida completo desde los puntos de vista técnico y de gestión"*[1] (en este caso, el modelo de procesos del ciclo de vida se limitaría a la provisión de un modelo para el proceso de MPE).

Entre los elementos con que debe contar una metodología, los autores citan:

1. Un modelo de procesos de ciclo de vida.
2. Un conjunto de técnicas.
3. Un conjunto de entregables.
4. Un conjunto de métricas.
5. Guías para la gestión del proyecto, incluyendo roles y estructura del equipo.
6. Herramientas.

Con esta definición, el concepto de metodología continúa siendo amplio pero encierra un significado mucho más preciso. Como se observa en la Figura 1.1, en ella hemos hecho la siguiente distinción dentro de las categorías de técnicas y métricas:

- Técnicas de estimación de recursos, de estimación de esfuerzo y de identificación y estimación de riesgos.

- Métricas de producto y métricas de proceso. Además, dentro de las primeras, están las métricas para bases de datos y para sistemas orientados a objetos.

[1] (Graham, Henderson-Sellers y Younessi (1997) definen el término en su libro "The OPEN Process Specification")

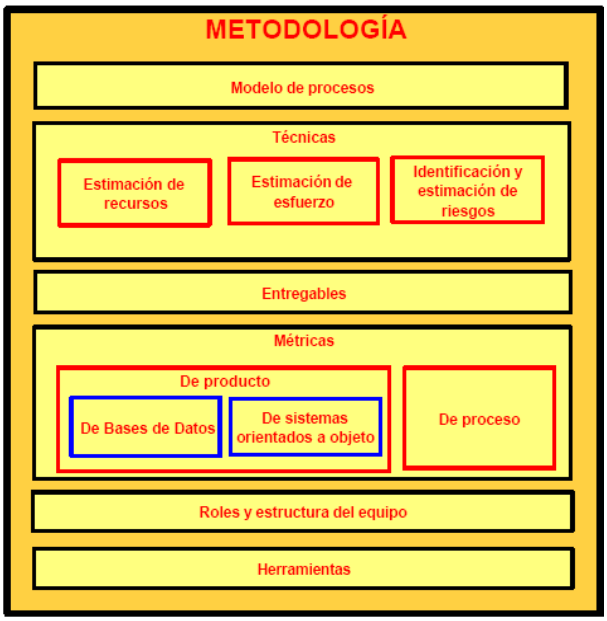


Figura 1.1 Componentes de una metodología[1]

[1](Graham, I., Henderson-Sellers, B. y Younessi, H. (1997). *The OPEN Process Specification*. Essex, Reino Unido: ACM Press y Addison-Wesley.)

2 CAPÍTULO II - MÉTODO DE TRABAJO

2.1 Investigación en Acción

En la edición de 1998 de la Conferencia sobre Procesamiento de Información se celebró la aceptación de métodos de investigación cualitativos como métodos de investigación apropiados para el campo de los sistemas de información (Avi son et al., 1999), y han merecido la atención de la revista *IEEE Transactions on Software Engineering* en su número especial sobre Ingeniería del Software Empírica (Seaman, 1999).

Investigación en acción *(Action research)* es un método de investigación cualitativo que McTaggart (1991) define como *"la forma que tienen los grupos de personas para preparar las condiciones necesarias para aprender de sus propias experiencias, y hacer estas experiencias accesibles a otros"*[1]. French y Bell (1996) matizan un poco más el concepto y lo definen como *"el proceso de recopilar de forma sistemática datos de la investigación acerca de un sistema actual en relación con algún objetivo, meta o necesidad de ese sistema; de alimentar de nuevo con esos datos al sistema; de emprender acciones por medio de variables alternativas seleccionadas dentro del sistema, basándose tanto en los datos como en las hipótesis; y de evaluar los resultados de las acciones, recopilando datos adicionales"*[2]. Según Wadsworth (1998), en Investigación en acción participan *"todas las partes involucradas en la investigación, examinando la situación existente (que sienten como problemática), con los objetivos de cambiarla y mejorarla"*[3]. Esta autora identifica los siguientes cuatro tipos de participantes en este método, que pueden coincidir en algunas ocasiones:

- El investigador, que es aquel que impulsa como sujeto el proceso investigador.
- El objeto investigador.

[1] (McTaggart, R. (1991). Principles of Participatory Action Research. *Adult Education Quarterly*, 41(3))
[2] (French, W.L. y Bell, C.H. Jr. (1996). *Desarrollo organizacional (quinta edición)*. Naucalpán de Juárez, México: Prentice-Hall.)
[3] (Wadswroth, Y. (1998). What is Participatory Action Research? *Action Research International*. Accedido el 10 de enero de 2000 en Internet: http://www.scu.edu.au/schools/sawd/ari/ari-wadsworth.html)

- Para quien se investiga, en el sentido de que tiene un problema que necesita ser resuelto y que participa en el proceso investigador. En este contexto se le denomina *grupo crítico de referencia*, y en él hay tanto personas que saben que están participando en la investigación, como otras que participan sin saberlo.

- Aquél para quien se investiga, en el sentido de que puede beneficiarse del resultado de la investigación, aunque no participe directamente en el proceso. Puede ser el receptor de documentos, informes, etc.

Chein, Cook y Harding[1] distinguen cuatro tipos de Investigación en acción:

- *De diagnóstico*, en la que el investigador se adentra en una situación problemática, la diagnostica y realiza recomendaciones al grupo crítico de referencia, pero sin que haya habido una evaluación de tales recomendaciones y sin un control posterior de su efecto.

- *Participante,* en la que el grupo de crítico de referencia pone en práctica las recomendaciones realizadas por el investigador, compartiendo con él sus efectos y resultados.

- *Empírica,* en la que el grupo crítico de referencia lleva un registro amplio y sistemático de sus acciones y sus efectos, característica que hace que esta variante sea difícilmente aplicable.

- *Experimental,* que consiste en evaluar las diferentes formas que existen para conseguir un objetivo. El principal inconveniente de esta variante reside, precisamente, en la dificultad de poder medir objetivamente las diferentes formas, ya que por lo general serán o bien aplicadas en distintas organizaciones con distintas características que enturbian los resultados de la investigación, o bien en una sola organización pero en distintos momentos, con lo que el entorno experimental habrá variado.

[1] (French, W.L. y Bell, C.H. Jr. (1996). *Desarrollo organizacional (quinta edición)*. Naucalpán de Juárez, México: Prentice-Hall.)

Para Padak y Padak (1994), el objeto de la investigación en acción es habitualmente un problema que necesita ser resuelto. Estos autores identifican los siguientes pasos, que deben seguirse en las investigaciones que utilicen este método:

- *Identificar las cuestiones que guiarán la investigación*, que deben ser relevantes, estar directamente relacionadas con el objeto que se está investigando y ser susceptibles de encontrarles respuesta.

- *Recoger información relacionada con tales cuestiones*, la cual puede proceder de prácticamente cualquier sitio (bibliografía, medidas, resultados de pruebas, observaciones, entrevistas, documentos, etc.).

- *Analizar la información recogida*, que debe realizarse cuando se pasa por un periodo durante el que no se obtiene información relevante.

- *Compartir los resultados con el resto de interesados*, de tal manera que se invite al planteamiento de nuevas cuestiones relevantes y, como añade Wadsworth (1998), *"a profundizar en la materia que se está investigando para proporcionar conocimientos nuevos que puedan mejorar las prácticas, modificando éstas como parte del propio proceso investigador, para luego volver a investigar sobre estas prácticas una vez modificadas"* [1].

Con estas características, el proceso de investigación definido por la Investigación en acción no es un proceso lineal, sino que va avanzando por ciclos, en cada uno de los cuales se ponen en marcha nuevas ideas, que son puestas en práctica y comprobadas hasta el siguiente ciclo (Wadsworth, 1998), como se muestra en la Figura 2.1.

[1] (Wadswroth, Y. (1998). What is Participatory Action Research? *Action Research International*. Accedido el 10 de enero de 2000 en Internet: http://www.scu.edu.au/schools/sawd/ari/ari-wadsworth.html)

Figura 2.1 Proceso cíclico de Investigación en Acción[1].

2.2 Método Seguido En La Propuesta De Métricas Y Técnicas

Hace poco tiempo, la validación de métricas se basaba fundamentalmente en la realización de experimentos con ellas, de manera que se pudiera demostrar o refutar su adecuación al propósito para el que hubieran sido construidas. No obstante, la aplicación de la teoría de la medida a la medición del software ha ido cobrando cada vez más importancia, habiéndose propuesto diferentes marcos formales para la caracterización teórica de las métricas de software.

En esta tesis se propone la utilización de algunas métricas para bases de datos y sistemas orientados a objetos que han sido caracterizadas, desde el punto de vista teórico, utilizando ciertos conjuntos de propiedades propuestos en la literatura científica. Esta "validación teórica" ha sido complementada en algunos casos con una validación empírica, en el sentido de que se han realizado experimentos para comprobar si las métricas resultan adecuadas para medir las propiedades del software que se pretenden. Este método de doble verificación se resume en la Figura 2.2: como se observa, los resultados de cualquiera de ambas validaciones pueden utilizarse para redefinir la métrica objeto del estudio.

[1] (Wadswroth, Y. (1998). What is Participatory Action Research? *Action Research International*. Accedido el 10 de enero de 2000 en Internet: http://www.scu.edu.au/schools/sawd/ari/ari-wadsworth.html)

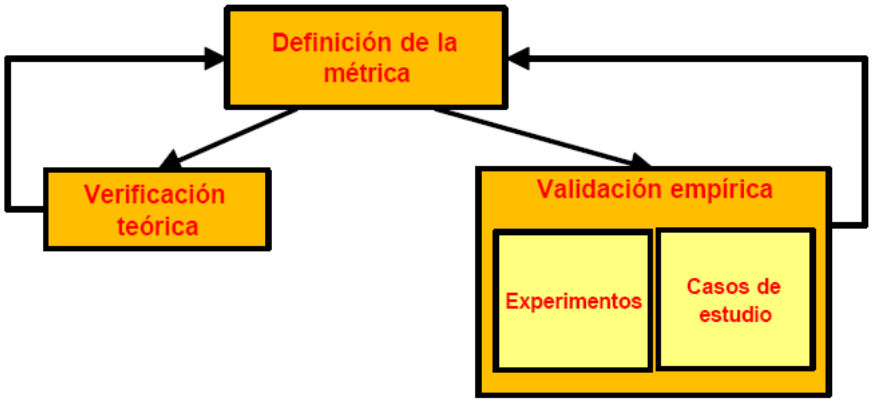


Figura 2.2 Verificación y validación de métricas[1]

[1] (Wadswroth, Y. (1998). What is Participatory Action Research? *Action Research International*. Accedido el 10 de enero de 2000 en Internet: http://www.scu.edu.au/schools/sawd/ari/ari-wadsworth.html)

3 CAPÍTULO III - ESTADO DEL ARTE

3.1 El Estándar IEEE 1074

Este estándar de ciclo de vida (IEEE, 1991) se puede aplicar a proyectos de desarrollo y mantenimiento de software. Consta de diecisiete procesos, cada uno compuesto de una serie de actividades.

El estándar puede utilizarse para cualquier ciclo de vida software; sin embargo, antes de hacer uso del estándar, deben realizarse correspondencias entre las actividades del estándar y las del modelo de ciclo de vida utilizado o desarrollado. Esta adaptación se realiza en el primer proceso (*Modelo del ciclo de vida software*). Tras ella, se ejecutan el proceso de *Inicio de proyecto* para preparar el marco de desarrollo del proyecto. A continuación, se llevan a cabo los procesos de predesarrollo, desarrollo y postdesarrollo.

Los procesos de *control y seguimiento del proyecto* y *gestión de la calidad software* son ejecutados paralelamente, durante toda la vida del proyecto. Por otra parte, se ejecutan los procesos que son necesarios para asegurar el cumplimiento satisfactorio de un proyecto, pero que no pueden ser considerados procesos de desarrollo:

- Verificación y validación
- Gestión de la configuración del software
- Desarrollo de la documentación
- Formación

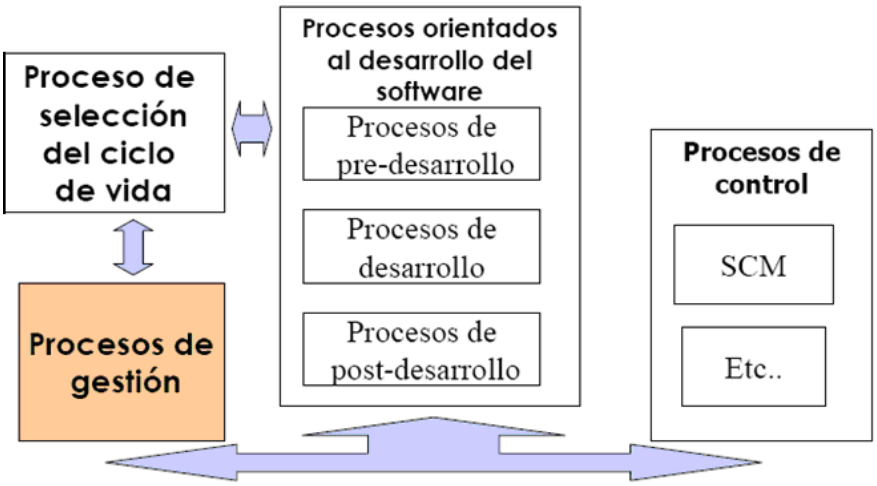


Figura 3.1 Gestión de Proyectos – IEEE 1074[1]

3.1.1 Gestión de proyectos

Los proyectos son un conjunto de actividades y recursos ordenados en el tiempo y que desarrollan un producto, bien o servicio. La gestión se centra en la coordinación de las actividades y recursos en el tiempo.

Las técnicas de gestión existentes no están orientadas a la gestión específica de proyectos de software pero pueden adaptarse.

Figura 3.2 Actividades del Proceso de Gestión del Proyecto[2]

[1] (IEEE (1991). IEEE Std. 1074-1995, IEEE Standard for Developing Software Life Cycle Processes Nueva York: The Institute of Electrical and Electronics Engineers, Inc.)
[2] (IEEE (1991). IEEE Std. 1074-1995, IEEE Standard for Developing Software Life Cycle Processes Nueva York: The Institute of Electrical and Electronics Engineers, Inc.)

3.1.2 La Planificación

La planificación es *"la elaboración de un plan general, científicamente organizado y frecuentemente de gran amplitud, para obtener un objetivo determinado, tal como el desarrollo económico, la investigación científica, el funcionamiento de una industria"*[1].

Importancia de la planificación de proyectos de software

- El software se desarrolla, no se fabrica en un sentido clásico.
- Un proyecto comienza con la estimación del trabajo a realizar, estimación de recursos necesarios, tiempo y costo de desarrollo.
- La estimación conlleva un grado de incertidumbre.
- Un buen planificador debe reducir esa incertidumbre.
- No existe modelo universal de estimación o fórmulas universalmente aplicables a todos los proyectos

3.1.3 El Tamaño y la Planificación

Dentro de un proyecto de software la etapa de planificación se considera importante, ya que se establece los objetivos y metas de los proyectos, las estrategias y políticas del proyecto.

Todo proyecto debe tener como base un plan, que ayudará al manejo de tareas, actividades, costos y tiempos necesarios para el cumplimiento del proyecto de software.

El manejador de costo para un proyecto es sin duda el tamaño del proyecto. La medida del tamaño debe estar relacionada directamente con el esfuerzo de desarrollo; por esta razón, las métricas de tamaño abarcan todos los aspectos que influyen en el costo, como tecnología, tipos de recursos y complejidad.

[1] (Norman Fenton, "Software measurement: A Necessary Scientific basis", IEEE Transactions on Software Engineering, vol 20, no 3, March 1994.)

Como soluciones para la disminución de incertidumbre en la estimación del tamaño del proyecto se considera el uso de registros históricos y esfuerzo requerido en proyectos anteriores.

3.2 Métricas y estimación del software

Las mediciones de software se utilizan para recolectar los datos cualitativos acerca del software y sus procesos para aumentar su calidad. La necesidad de la medición es algo evidente. Después de todo es lo que nos permite cuantificar y por consiguiente gestionar de forma más efectiva. Pero la realidad puede ser muy diferente.

Las métricas nos ayudan a entender tanto el proceso técnico que se utiliza para desarrollar un producto, como el propio producto.

Hay varias razones para medir:

1. Para indicar la calidad del producto.
2. Para evaluar la productividad de la gente que desarrolla el producto.
3. Par evaluar los beneficios en términos de productividad y de calidad, derivados del uso de nuevos métodos y herramientas de la ingeniería de software.
4. Para establecer una línea de base para la estimación.
5. Para ayudar a justificar el uso de nuevas herramientas o de formación adicional.

3.2.1 Estimación del Software

"Es una pequeña planeación sobre el desarrollo futuro del proyecto. Una de las actividades fundamentales del proceso de gestión del proyecto de software es la planificación. Cuando se planifica un proyecto de software se obtiene estimaciones de la duración cronológica del esfuerzo humano requerido, del tiempo y del costo"[1].

[1] (Fenton, N. E. y Pfleeger, S. L. (1997). Software Metrics. A Rigorous & Practical Approach. International Thomson Computer Press)

Se han desarrollado varias técnicas de estimación para el desarrollo de software, aunque cada una tiene sus puntos fuertes y sus puntos débiles, todas tienen en común los siguientes atributos:

1. Establecer el ámbito del proyecto.
2. Como fundamento para la realización de estimaciones se usan métricas del software de proyectos pasados.
3. El proyecto se divide en partes más pequeñas que se estiman individualmente.

3.2.2 Experiencia con la Estimación del Tamaño del Software

La siguiente experiencia en la estimación del tamaño de un proyecto de software, está basado en una especificación de requisitos bastante completa.

Experiencia: Se desarrolló dentro de la asignatura Gestión de Proyectos de Ingeniería de Software. Esta experiencia consistió en un simulacro de llamado a licitación para el desarrollo de un producto software para una organización ficticia.

Tareas: Los alumnos debían determinar el tamaño del producto, estimar costos y plazos. La mayoría de ellos escogió la métrica puntos de función por sobre otras (puntos de característica, líneas de código) por las ventajas que esta métrica tiene.

Técnica utilizada: Todos los alumnos manejaban la técnica de conteo de puntos de función, y utilizaron el mismo método (guía) para realizar el conteo.

Desarrollo: Todos los alumnos debieron entregar el detalle de las tareas desarrolladas como anexo a la propuesta, por lo que se pudo verificar la correcta aplicación del método para el caso de la estimación del tamaño del software a desarrollar.

Análisis: Se presenta una breve visión de la influencia de la determinación del tamaño en la planificación de un proyecto, además de una revisión a la importancia de las estimaciones en etapas tempranas del desarrollo.

3.2.3 Estimación Temprana del Tamaño del Software

Para la planificación de un proyecto de software es importante realizar una estimación certera del esfuerzo necesario para el desarrollo; difícilmente se puede realizar una estimación certera de la cantidad de líneas de código que tendrá la aplicación, ya que en este nivel no tiene por qué estar decidida la herramienta de desarrollo y la cantidad de líneas de código para implementar una función del software dependen fuertemente del programador y la herramienta de desarrollo.

Los puntos de función pueden ser estimados con mayor certeza a partir de una buena especificación de requisitos.

Lamentablemente, cuando los proyectos se plantean sobre la base de especificaciones vagas o inexistentes.

3.2.4 Resultados de una Experiencia en la Determinación del Tamaño de un Software

Para presentar una respuesta a un llamado a licitación, 13 alumnos de quinto año debieron determinar una buena estimación del tamaño del producto a desarrollar, eligiendo 10 de ellos la métrica puntos de función.

Los resultados obtenidos se muestran a continuación.

Resultados: La medida en puntos de función obtenida por cada alumno, están en un rango de 181 a 759.

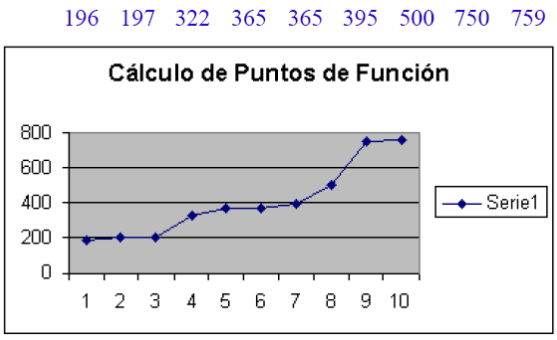


Figura 3.3 Cuadro de Resultados de Experiencia de Estimación

La desviación estándar de estos datos es de aproximadamente 211 PF (Puntos de Función) sobre el promedio, lo que es sin duda muy alto. Esta gran diferencia condujo a estimaciones de esfuerzo y plazos muy distintas. Esto influyó directamente en la cantidad de personal que cada alumno asignó al proyecto y a la organización del mismo, además en los costos y plazos.

Durante el desarrollo del proyecto, las diferencias en este tipo de estimaciones causan variaciones en la dirección, afectando al proyecto en su totalidad, y por lo tanto al éxito o fracaso del mismo.

3.2.5 Explicación de Resultados

La asignación de complejidad que el método exige para cada ítem (entrada, salida, consulta, archivo lógico, archivo de interfaz), es subjetiva y difiere de un desarrollador a otro, por los diversos criterios y experiencia dentro de los proyectos de software, pero esto no explica el resultado obtenido en desviación estándar (211PF).

El proyecto de software que fue objeto de medición, está organizado de acuerdo al tipo de información. Por ejemplo, los reportes y consultas están claramente especificados, mediante un diseño de formularios, pero no así las entradas. Sólo se especifican los datos que se deben ingresar. Es decir, que el número de salidas y consultas estaban bien definidos, lo que no sucedía con las entradas y archivos lógicos internos.

Las entradas fueron agrupadas de acuerdo al criterio de los alumnos, mientras que otros alumnos consideraron que cada dato para ingresar era una entrada; lo que fue motivo para aumentar la cantidad en el número de entradas y superar el número de consultas y salidas. A pesar de que la complejidad de las entradas que involucran a más de un dato es mayor que la de una entrada de un solo dato, esto no alcanzó a corregir la diferencia entre las medidas.

Para la determinación de los archivos lógicos internos, algunos alumnos agruparon grandes conjuntos de información en un archivo lógico interno, mientras que otros modelaron la información en un esquema entidad relación, obteniendo una aproximación más cercana del número de tablas a implementar.

Otro factor que dificulta el cálculo del tamaño del software es el criterio personal que tiene cada alumno para considerar cada requisito del proyecto como entrada, salida, archivo lógico, consulta.

Otro factor importante, es la presión a la que estaban sujetos los participantes. Debían entregar una propuesta de calidad dentro de los plazos, o su castigo sería equivalente a no ganarse el proyecto: una baja calificación, o peor aún, reprobar la asignatura.

3.2.6 Conclusión

No se debe realizar una estimación temprana de proyectos porque la obtención de resultados no se basa en una técnica específica, ni considera como base la Especificación de Requisitos del sistema de software a desarrollar.

3.2.7 Tipos de Métricas de Software

Las métricas están relacionadas con el desarrollo del proyecto de software, considerando la funcionalidad, complejidad y eficiencia en las etapas de desarrollo (Universidad de Guadalajara, Otoniel Pérez Giraldo, 1998).

- ***MÉTRICAS TÉCNICAS***

  Se centran en las características de software por ejemplo: la complejidad lógica. Mide la estructura del sistema, y el cómo esta hecho.

- ***MÉTRICAS DE CALIDAD***

  Proporcionan una indicación de cómo se ajusta el software a los requisitos implícitos y explícitos del cliente. Es decir cómo se va a medir para que el sistema se adapte a los requisitos que solicita el cliente.

- ***MÉTRICAS DE PRODUCTIVIDAD***

  Se centran en el rendimiento del proceso de la ingeniería del software. Es decir que tan productivo va a ser el software a diseñar.

- ***MÉTRICAS ORIENTADAS A LA PERSONA***

  Proporcionan medidas e información sobre la forma que la gente desarrolla el software y sobre todo el punto de vista humano de la efectividad de las herramientas y métodos.

- ***MÉTRICAS ORIENTADAS AL TAMAÑO***

  Permite conocer en qué tiempo se terminará el software y cuántas personas se necesitarán para el desarrollo del proyecto. Son medidas directas al software y al proceso por el cual se desarrolla, si una organización de software mantiene registros sencillos, se puede crear una tabla de datos orientados al tamaño como se muestra en la siguiente figura:

Tabla 3.1 Ejemplo de Proyectos de Software

PROYECTO	ESFUERZO	$	KLDC	PAGS. DOC	ERRORES	GENTE
999-01	24	168	12.1	365	29	3
CCC-04	62	440	27.2	1124	86	5
FFF-03	43	314	20.2	1050	64	6
.	.	.	.	.	.	.
.	.	.	.	.	.	.

La tabla lista cada proyecto del desarrollo del software de los últimos años correspondientes, datos orientados al tamaño de cada uno. Refiriéndonos a la entrada de la tabla del proyecto 999-01 se desarrollaron 12.1 KLDC[1] con un esfuerzo de 24 personas mes y un costo de 168 mil dólares. Debe tenerse en cuenta que el esfuerzo y el costo registrados en la tabla incluyen todas las actividades de la ingeniería de software como son análisis, diseño, codificación y prueba. Otra información del proyecto 222-01 indica que se desarrollaron 365 páginas mientras que se encontraron 29 errores tras entregárselo al cliente, dentro del primer año de utilización también sabemos que trabajaron 3 personas en el desarrollo del proyecto.

[1] (Miles de Líneas de Código)

Fórmulas:

Productividad = KLDC/persona-mes

Calidad = errores/KLDC

Documentación = pags. Doc/ KLDC

Costo = $/KLDC

Persona-mes = esfuerzo

- ***MÉTRICAS ORIENTADAS A LA FUNCIÓN***

  Las métricas orientadas a la función fueron propuestas por Albrecht, quien sugirió un acercamiento a la medida de la productividad denominado método del punto de función. Los puntos de función se obtienen utilizando una función empírica basando en medidas cuantitativas del dominio de información del software y valoraciones subjetivas de la complejidad del software.

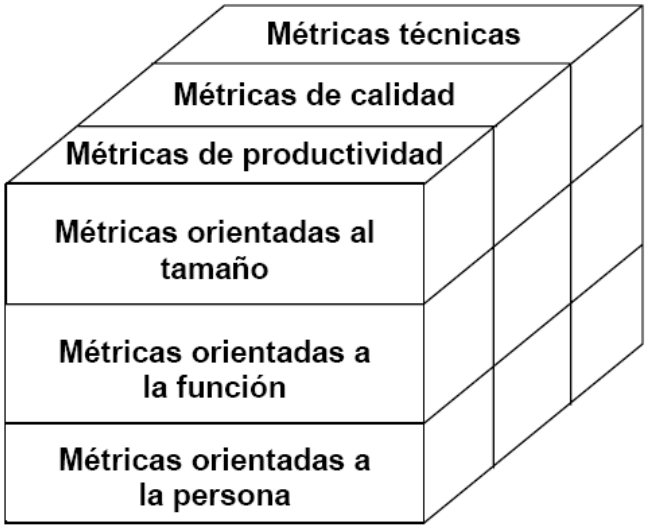


Figura 3.4 Clasificación de las métricas del software

3.2.8 Técnicas de Estimación

3.2.8.1 Métrica de Puntos de Función

Justificación

Las métricas de punto de función, pretende medir la funcionalidad entregada al usuario independientemente de la tecnología utilizada para la construcción y explotación del

software; y también ser útil en cualquiera de las etapas dentro del ciclo de vida del software, desde la obtención de requisitos hasta la explotación y mantenimiento.

Las métricas de puntos de función son medidas indirectas del software y del proceso por el cual se desarrolla. En lugar de calcular las LDC[1], las métricas orientadas a la función se centran en la funcionalidad o utilidad del programa.

La utilización del Modelo Entidad-Relación como base para el cálculo de los puntos de función, sirve de guía para determinar la cantidad de entradas, salidas externas, consultas, archivos lógicos e interfaces que utilizará el proyecto de software; siendo un método diferente en el cálculo de estimación usual. Esto hace que ambos modelos, Puntos de función y Modelo Entidad-Relación, sean compatibles y complementarios.

Antecedentes

Tradicionalmente se ha medido el tamaño del software mediante el conteo de las líneas de código, número de programas fuente, o técnicas similares, que no resultan aceptables como una buena práctica profesional, por diferentes razones:

- Su resultado depende fuertemente del entorno técnico y el lenguaje de programación utilizado.
- Varía en función de la técnica que utiliza cada programador y del uso de normas y metodologías.
- No resultan significativas al usuario ni a la dirección.

Cuando se trata de establecer métricas de productividad y calidad en la construcción de software, o realizar estimaciones de coste y duración, es imprescindible disponer de una medida fiable y clara del tamaño de lo que se construye.

Normalización

La organización ISO/IEC[2] ha definido un estándar de Medida del Tamaño Funcional, titulado 'ISO/IEC 14143-1:1998'. Con base en este estándar se han declarado los siguientes métodos estándares:

[1] (Líneas De Código)
[2] (International Standard Organization /International Electrotechnical Commission)

- ISO/IEC 20926:2003 IFPUG[1] 4.1 Método de medición del tamaño functional sin ajustar – Conteo práctico manual (Unadjusted functional size measurement method - Counting practices manual).
- ISO/IEC 19761:2003 COSMIC-FFP – Método de medición del tamaño functional (A Functional Size Measurement Method).
- ISO/IEC 20968:2002 Mk II Análisis de puntos de function – Conteo práctico manual (Function Point Analysis - Counting Practices Manual).
- ISO/IEC 24570:2004 NESMA Guía usando el análisis de puntos de función (Guide to Using Function Point Analysis).

La norma española equivalente a la ISO 14143 es la UNE 71045-1:2000[2].

Elementos de Función y Clasificación

Se determinan 5 elementos de función y están definidos de la siguiente manera:

- Entradas Externas
- Salidas Externas
- Consultas Externas
- Ficheros Lógicos Internos
- Ficheros Externos de Interfaz

- **Entradas Externas (EI, External Input):**

  Permite la entrada de datos al sistema. Estos datos provienen bien de una aplicación ajena al sistema, o bien del usuario, el cual los introduce a través de una pantalla de entrada de datos (órdenes concretas, nombres de ficheros, selecciones de menús, etc.). No se incluyen las consultas o peticiones interactivas al sistema, ya que éstas se contabilizan por separado. Los datos de entrada son usados para mantener uno o más Ficheros Lógicos Internos, siempre y cuando no representen información de control del sistema. Para determinar las Entradas Externas, se suelen examinar las pantallas de introducción de datos, los

[1] (International Function Point Users Group)
[2] (Tecnología de la información. Medida del software. Medida del tamaño funcional. Parte 1: Definición de conceptos)

cuadros de diálogo y el formato de los formularios de entrada, si es que existen. Además, si se trata de entradas procedentes de otras aplicaciones distintas, éstas deberán necesariamente actualizar los Ficheros Lógicos Internos del sistema que se pretende medir. Como consecuencia de una entrada, siempre deberá actualizarse un fichero lógico interno.

Tabla 3.2 Definición de Complejidad para EI

DIFICULTAD ENTRADAS	Número de Campos o Atributos de la Entrada		
	1-4 Atributos	5-15 Atributos	16 + Atributos
0 ó 1 ficheros accedidos	BAJA	BAJA	MEDIA
2 ficheros accedidos	BAJA	MEDIA	ALTA
3 + ficheros accedidos	MEDIA	ALTA	ALTA

- **Salidas Externas (EO, External Output):**

  Permite la salida de datos del sistema. Estos datos suelen ser los resultados de algoritmos o la evaluación de fórmulas, y generan informes (reports) o archivos de salida que sirven de entrada a otras aplicaciones. En la creación de estos informes o archivos de salida intervienen uno o más Ficheros Lógicos Internos o uno o más Ficheros Externos de Interfaz. Las salidas se refieren a informes, pantalla y mensajes de error.

Tabla 3.3 Definición de Complejidad para EO

DIFICULTAD SALIDAS	Número de Campos o Atributos de la Salida		
	1-5 Atributos	6-19 Atributos	20 + Atributos
0 ó 1 ficheros accedidos	BAJA	BAJA	MEDIA
2 ó 3 ficheros accedidos	BAJA	MEDIA	ALTA
4 + ficheros accedidos	MEDIA	ALTA	ALTA

- **Consultas Externas (EQ, External Query):**

  Consiste en la selección y recuperación de datos de uno o más Ficheros Lógicos Internos o de uno o más Ficheros Externos de Interfaz, y su posterior devolución al usuario o aplicación que los solicitó. Son peticiones interactivas que requieren una respuesta del sistema. En el proceso de entrada no se actualiza ningún Fichero Lógico Interno, y en el proceso de salida los datos devueltos no

contienen datos derivados (es decir, datos resultantes de la ejecución de algoritmos o la evaluación de fórmulas). Una posible forma de detectar las Consultas Externas es examinando los formularios de entrada, las pantallas de entrada de datos, los cuadros de diálogo, etc.

Tabla 3.4 Definición de Complejidad para EQ

DIFICULTAD SALIDAS	Número de Campos o Atributos		
	1-5 Atributos	6-19 Atributos	20 + Atributos
0 ó 1 ficheros accedidos	BAJA	BAJA	MEDIA
2 ó 3 ficheros accedidos	BAJA	MEDIA	ALTA
4 + ficheros accedidos	MEDIA	ALTA	ALTA

- **Ficheros Lógicos Internos (ILF, Internal Logic File):**

  Un Fichero Lógico Interno es un conjunto de datos definidos por el usuario y relacionados lógicamente, que residen en su totalidad dentro de la propia aplicación, y que son mantenidos a través de las Entradas Externas del sistema. Para determinar los posibles Ficheros Lógicos Internos se suelen examinar los modelos físicos y/o lógicos preliminares, los formatos de tablas, las descripciones de bases de datos, etc. Los ficheros se cuentan una sola vez, independientemente del número de procesos que los acceden.

Tabla 3.5 Definición de Complejidad para ILF

DIFICULTAD FICHEROS LÓGICOS	Número de Campos o Atributos		
	1-19 Atributos	20-50Atributos	51 + Atributos
1 Registro Lógico	BAJA	BAJA	MEDIA
2 a 5 Registros Lógicos	BAJA	MEDIA	ALTA
6 o más Registros Lógic.	MEDIA	ALTA	ALTA

- **Ficheros Externos de Interfaz (EIF, External Interface File):**

  Un Fichero Externo de Interfaz es un conjunto de datos definidos por el usuario, que están relacionados lógicamente y que sólo son usados para propósitos de referencia. Los datos residen en su totalidad fuera de los límites de la aplicación y son mantenidos por otras aplicaciones. En definitiva, un Fichero Externo de

Interfaz es un Fichero Lógico Interno para otra aplicación. Para determinar los posibles Ficheros Externos de Interfaz se suelen analizar las descripciones de interfaces del sistema con otras aplicaciones. Se cuentan todas las interfaces legibles por la máquina por ejemplo: archivos de datos, en cinta o discos que son utilizados para transmitir información a otro sistema.

Tabla 3.6 Definición de Complejidad para EIF

DIFICULTAD FICHEROS DE INTERFAZ	Número de Campos o Atributos		
	1-19 Atributos	20-50Atributos	51 + Atributos
1 Entidad o Registro Lógico	BAJA	BAJA	MEDIA
2 a 5 Registros Lógico	BAJA	MEDIA	ALTA
6 o más Registros Lógic.	MEDIA	ALTA	ALTA

Agrupando las Entradas Externas, las Salidas Externas y las Consultas Externas se tiene el conjunto de **Transacciones Lógicas** del sistema, y agrupando los Ficheros Lógicos Internos y los Ficheros Externos de Interfaz se tiene el conjunto de **Ficheros Lógicos** del sistema.

Cálculo de PFSA, PFA y Esfuerzo Requerido

Con los elementos de función definidos anteriormente, se calcula los **puntos de función sin ajustar (PFSA)**. Posteriormente, se aplica el **factor de ajuste (VFA)** que resulta de valoraciones subjetivas efectuadas a la aplicación que está siendo medida y de su entorno. La suma de los puntos de función sin ajustar y el factor de ajuste representan los **puntos de función ajustados (PFA)**.

A continuación se muestra en detalle los valores que se deben considerar para el cálculo de: puntos de función sin ajustar, factor de ajuste y puntos de función ajustados.

Puntos de Función Sin Ajustar (PFSA):

Tabla 3.7 Valoración de Complejidad de los Elementos de Función

	Simple		Media		Compleja		Total
	Cantidad	* Peso	Cantidad	* Peso	Cantidad	* Peso	
Entradas		* 3		* 4		* 6	
Salidas		* 4		* 5		* 7	
Consultas		* 3		* 4		* 6	
Fic. Lógicos		* 7		* 10		* 15	
Fic. Interfaz		* 5		* 7		* 10	
Total puntos de función sin ajustar (PFSA)							

Tabla 3.8 Las 14 características generales del sistema, para el cálculo de Puntos Función[1]

Características Generales del Sistema (GSC's)		Cuestiones	Ejemplo
1	Comunicación de datos	¿Qué necesidades de comunicación requiere el sistema para transferencia o intercambio de información?	Una aplicación para el sector bancario, donde se requieren numerosas transacciones monetarias.
2	Procesamiento de datos distribuido	¿Existen funciones de procesamiento distribuido? ¿Cómo son manejados los datos distribuidos?	Un motor de búsqueda en Internet, donde el procesamiento está distribuido en decenas de máquinas.
3	Rendimiento	¿Es importante el tiempo de respuesta? ¿Es crítico el rendimiento?	Una aplicación para el control del tráfico aéreo, que debe proporcionar continuamente información precisa sobre la posición y rumbo de los aviones.
4	Uso del hardware existente	¿En qué medida se está utilizando la plataforma hardware en donde se ejecutará la aplicación?	Un sistema para matrículas en una universidad, donde concurren cientos de alumnos al mismo tiempo.
5	Transacciones	¿Con qué frecuencia se ejecutan las transacciones? (diariamente, semanalmente, mensualmente, etc...)	Una aplicación para el sector bancario, donde deben realizarse millones de transacciones durante la noche.
6	Entrada de datos interactiva	¿Requiere el sistema entrada de datos interactiva? ¿Cuánta información se captura on-line? (en %)	Un programa en el que los datos de entrada provienen de papeles o formularios impresos.
7	Eficiencia	¿Se diseñó la aplicación pensando en que fuera eficiente y fácilmente utilizable por el usuario?	Un programa de análisis financiero utilizado por el directivo de una empresa, capaz de orientarle y asesorarle.
8	Actualizaciones on-line	¿Cuántos ficheros Ficheros Lógicos Internos se actualizan interactivamente (por medio de transacciones on-line)?	Una aplicación para reserva de billetes, en la que deben bloquearse y modificarse ciertos registros en las BB.DD.'s para evitar que un mismo asiento sea vendido dos veces.
9	Complejidad de procesamiento	¿Existe mucha carga en cuanto a procesami. lógico y/o matemático? ¿Es complejo el procesamiento interno?	Un sistema para diagnóstico médico, el cual realiza costo-sas operaciones de decisión lógica hasta obtener un result.
10	Reusabilidad	¿Se desarrolló la aplicación para cumplir las necesidades de más de un usuario? ¿Se ha diseñado el código para ser reutilizable?	Un procesador de textos en el que, por ejemplo, su barra de menús puede utilizarse desde una hoja de cálculo, un generador de informes de una base de datos, etc...
11	Facilidad de conversión e instalación	¿Cómo son de difíciles la conversión y la instalación? ¿Se ha incluido en el diseño la conversión y la instalación?	Cualquier aplicación de propósito general, de tal forma que cualquier persona pueda realizar la instalación fácilmente.
12	Facilidad de operación	¿Requiere el sistema copias de seguridad y de recuperación fiables? ¿Cómo son de efectivos y qué grado de automatización tienen los procesos de arranque, copia de seguridad y recuperación de datos?	Una aplicación para tratamiento de grandes cantidades de información, donde es muy importante la efectividad de los procesos de backup y recuperación de datos.
13	Múltiples instalaciones	¿Se diseñó y desarrolló el sistema para soportar múltiples instalaciones en diferentes organizaciones?	Una aplicación software para una multinacional con oficinas en varios países.
14	Facilidad de mantenimiento	¿Se diseñó y desarrolló el sistema pensando en facilitar el posterior proceso de mantenimiento?	

Factor de complejidad Técnica o Valor de Ajuste (VAF):

El factor de complejidad se lo utiliza para valorar las 14 características generales del sistema para el cálculo de los Puntos de Función Ajustados.

[1] (Documento de Planificación y Gestión de Sistemas de Información, "Medida del Tamaño Funcional de Aplicaciones Software", por Faustino Sánchez Rodríguez, Mayo 1999, pág. 13)

Tabla 3.9 Valores de los factores de complejidad

Valor	Significado del valor
0	Sin influencia, factor no presente
1	Influencia insignificante, muy baja
2	Influencia moderada o baja
3	Influencia media, normal
4	Influencia alta, significativa
5	Influencia muy alta, esencial

Cálculo de los PFA:

PFA = PFSA * (0,65 + (0.01 * VAF))

Cada factor de complejidad afecta en +/- 2,5% en los PFSA

PFSA * 65% <= PFA <= PFSA * 135%

Tabla 3.10 Conteo de Puntos de Función

Componente	Complejidad del componente (factor de peso)			Total
	Baja	Media	Alta	
Entradas Externas	____ x 3 = ____	____ x 4 = ____	____ x 6 = ____	____
Salidas Externas	____ x 4 = ____	____ x 5 = ____	____ x 7 = ____	____
Consultas Externas	____ x 3 = ____	____ x 4 = ____	____ x 6 = ____	____
Ficheros Lógicos Internos	____ x 7 = ____	____ x 10 = ____	____ x 15 = ____	____
Ficheros Externos de Interfaz	____ x 5 = ____	____ x 7 = ____	____ x 10 = ____	____
			Nº Total de Puntos Función sin Ajustar (PFsA):	____
			Factor de Ajuste (VAF):	x ___
			Nº Total de Puntos Función Ajustados (PFA):	_____

3.2.8.2 Métrica Bang

En las primeras etapas del desarrollo de un proyecto de software se realizan predicciones y estimaciones acerca del coste y la duración de procesos. Pero este tipo de estimaciones requiere (más aún en las primeras etapas) que hayan sido totalmente identificados los requisitos de usuario y que las especificaciones de dichos requisitos hayan quedado perfectamente definidas, para entender el alcance y la complejidad del sistema software al que el equipo de desarrollo se enfrenta. Para conseguir esto, se utiliza un modelo compuesto del sistema, obtenido de la unión de otros tres modelos: el

modelo funcional (qué hace el sistema), el modelo de datos (qué datos utiliza) y el modelo de comportamiento (cómo se comporta el sistema). Esta solución permitirá obtener una representación gráfica, concisa y fácil de entender cuáles son las especificaciones del sistema.

El modelo de requisitos de software está altamente estructurado, y representa la funcionalidad del sistema. Las métricas de especificación, entre las que se encuentra la métrica Bang, serán las encargadas de captar y medir dicha funcionalidad.

Definición de Métrica Bang

Tom DeMarco (1982) dice que *"es una métrica orientada a la función, que indica el tamaño funcional de un sistema. Para su utilización es necesario determinar el tamaño del modelo de especificación o modelo de requisitos del software"*[1].

Componentes elementales del modelo de especificación

Un componente del modelo de especificación se considera elemental (o primitivo) si no puede dividirse en otros más pequeños. Cada parte del modelo de especificación (modelo funcional, modelo de datos y modelo de comportamiento) se divide y descompone reiteradamente hasta llegar al nivel más bajo de descomposición, donde todos los componentes son elementales.

Existen seis tipos distintos de componentes:

1. Primitivas funcionales: Son los componentes elementales que se derivan del modelo funcional (descomposiciones de los procesos de un diagrama de flujo de datos, por ejemplo). Ejemplo de primitiva funcional es cualquier proceso indivisible que transforma un dato de entrada en un dato de salida. Las primitivas funcionales forman parte del modelo funcional.

2. Datos elementales: Son los datos concretos e indivisibles que forman parte del diccionario de datos del modelo funcional. Datos elementales son: números, variables, cadenas, etc.

[1] (Tom DeMarco (1982), Methodology for software metrics)

3. Objetos: Son componentes que se derivan del modelo de datos (descomposiciones de un diagrama E/R, por ejemplo). Un objeto es, conceptualmente, un conjunto de datos caracterizados por los mismos atributos, y que constituyen una entidad.

4. Interrelaciones: Son componentes derivados de las relaciones (conexiones) existentes entre objetos. Forman parte del modelo de datos del sistema.

5. Estados: Son componentes elementales que se derivan del modelo de comportamiento (descomposiciones de un diagrama de estados/transiciones, por ejemplo). Lógicamente, los estados forman parte del modelo de comportamiento.

6. Transiciones: Son las acciones que hacen pasar al sistema de un estado a otro del modelo de comportamiento.

Contando el número de componentes elementales que aparece en el modelo de especificación del sistema, obtenemos los datos de partida para utilizar la métrica *Bang*.

DeMarco (1982) añade otros seis tipos de componentes más. Los mismos que se muestra en la tabla 3.11:

Tabla 3.11 Componentes elementales que han de contabilizarse para aplicar la métrica Bang[1]

Componente elemental:	Descripción:
FP	Número de **primitivas funcionales** del modelo funcional del sistema.
FPM	Número de **primitivas funcionales modificadas** (primitivas funcionales externas al sistema que deben modificarse para adaptarlas al nuevo sistema).
DE	Número de **datos elementales** del sistema.
DEI	Número de **datos elementales de entrada** al sistema.
DEO	Número de **datos elementales de salida** del sistema.
DER	Número de **datos elementales** ***retenidos*** por el sistema (datos almacenados dentro del sistema).
OB	Número de **objetos** del modelo de datos del sistema.
RE	Número de **interrelaciones** del modelo de datos del sistema.
ST	Número de **estados** del modelo de comportamiento del sistema.
TR	Número de **transiciones** del modelo de comportamiento del sistema.
TC_i	Número de ***tokens*** implicados en la *i-ésima* primitiva funcional (tanto de entrada como de salida). Un *token* es un dato o conjunto de datos que la primitiva funcional trata como un todo, es decir, no requiere ser dividido para que la primitiva funcional pueda utilizarlo. El parámetro TC_i debe calcularse para cada primitiva funcional.
RE_i	Número de interrlaciones en las que está involucrado el *i-ésimo* objeto del modelo de datos. Este parámetro debe calcularse para cada objeto del modelo de datos.

[1] (Documento "Planificación y Gestión de Sistemas de Información, Medida del Tamaño Funcional de Aplicaciones Software", por Faustino Sánchez Rodríguez, Mayo 1999, pág. 38)

A la hora de contar el número de componentes elementales en el modelo de especificación, se debe comprobar que el modelo no tenga redundancias entre sus componentes.

Clasificación de los sistemas

Luego que se ha contado los componentes elementales del modelo de especificación, se deberá elegir uno de ellos como indicador principal (sobre el que se basará el cálculo del tamaño funcional) y utilizar los otros como modificadores. Para la mayoría de los sistemas, este indicador principal suele ser el componente FP, es decir, las primitivas funcionales del sistema.

El factor FP es un indicador bastante preciso de la medida del tamaño funcional de la aplicación, por esta razón en la mayoría de los casos en los que el sistema en estudio tenga un alto componente funcional, es decir se de mayor importancia al DFD[1], se utilizará como indicador principal al factor FP.

Existen casos en los que el sistema está fuertemente orientado a los datos. En estos casos el indicador principal será el factor OB.

Se proponen tres criterios para clasificar los sistemas:

- Si **RE/FP < 0.7** entonces el sistema está fuertemente orientado a proceso y transformación de datos.
- Si **RE/FP > 1.5** entonces el sistema está fuertemente orientado a los datos.
- Si **(RE/FP ≥ 0.7) Y (RE/FP ≤ 1.5)** entonces el sistema es *híbrido*, es decir, se trata de un sistema con un importante componente funcional y de datos.

RE es el número de interrelaciones del modelo de datos del sistema, y **FP** es el número de primitivas funcionales del modelo funcional del sistema.

El factor **DEO** es el número de datos elementales de salida del sistema.

[1] (Diagrama Funcional de Datos)

Ratio DEO/FP: es un ratio indicativo de los datos de dentro hacia fuera del sistema. Los sistemas comerciales o de gestión tienen un ratio DEO/FP muy elevado, mientras que los sistemas científicos o de ingeniería tienen un ratio DEO/FP menor. No existe un valor concreto del ratio.

La siguiente figura muestra cómo pueden catalogarse los sistemas software, en función de estos dos ratios:

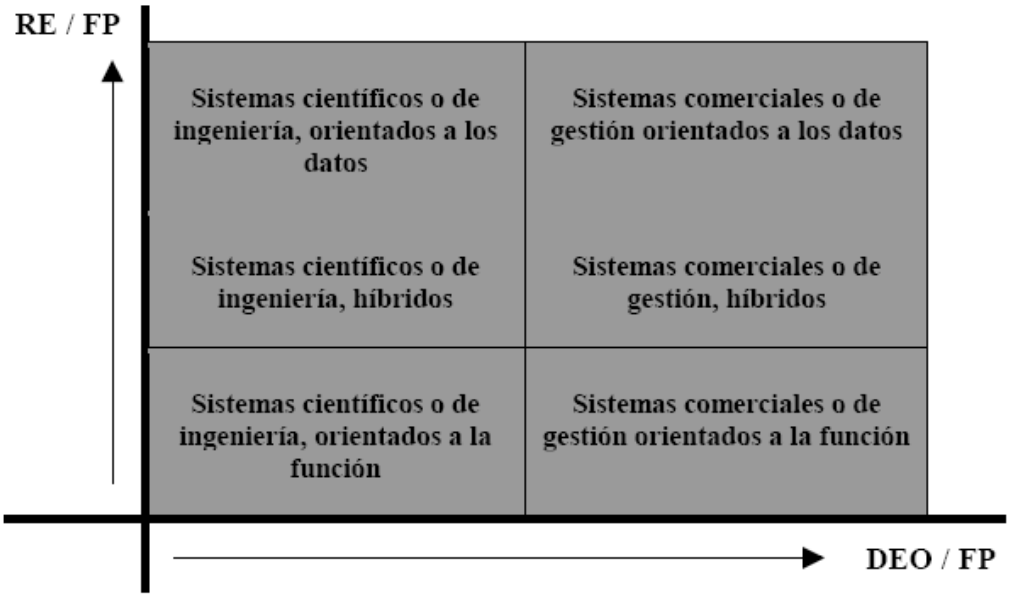


Figura 3.5 Catalogación de los Sistemas software, en función de los ratios DEO/FP y RE/FP[1]

Formulación de la métrica Bang

Como se ha visto anteriormente se distingue tres tipos de sistemas software: los orientados a los datos, los orientados a la función (proceso y transformación de esos datos) y los híbridos.

[1] Documento "Planificación y Gestión de Sistemas de Información, Medida del Tamaño Funcional de Aplicaciones Software", por Faustino Sánchez Rodríguez, Mayo 1999, pág. 39.

Por lo tanto, la métrica Bang se formula de tres formas distintas:

A. Para sistemas orientados a la función

En los sistemas orientados a la función, el componente elemental más importante son las primitivas funcionales. Durante la construcción del modelo de especificación (modelo funcional), existe el problema al no saber cuando dividir un proceso en otros subprocesos, o cuando dejarlo como proceso elemental. Este hecho hace que las personas tomen soluciones distintas. Por esta razón, al final, la contabilización de primitivas funcionales (factor FP), variará en unos casos y en otros. Teniendo como fundamento la siguiente regla a seguir:

"En el proceso de descomposición de un modelo, dejaremos un componente como primitivo o elemental, siempre que no sea posible su división o descomposición, o siempre que su división o descomposición no reduzca el factor TCmedio"[1].

El factor *TCmedio* a que se refiere viene dado por:

$$TC_{medio} = \frac{\sum TC_i}{FP}$$

TCmedio es el número medio de tokens[2] por primitiva funcional.

Para evitar que una primitiva funcional sea más grande que otra, se necesita un método de corrección. Para ello, deberemos entender una primitiva funcional como un proceso elemental en el que se transforman unos datos de entrada en otros de salida. El factor a utilizar será el TCi, es decir, el número de tokens involucrados en la i-ésima primitiva funcional, bien sean de entrada a la primitiva o de salida.

Conociendo entonces el valor de TCi para cada primitiva funcional, podemos aplicar la siguiente regla, para conocer cuál sería el tamaño corregido de cada primitiva funcional (CFPIi):

El tamaño de la primitiva "i" es proporcional a $TC_i * \log_2(TC_i)$

[1] (Tom DeMarco (1982), Methodology for software metrics)

[2] (Un token es un dato o conjunto de datos que la primitiva funcional trata como un todo, es decir, no requiere ser dividido para que la primitiva funcional pueda utilizarlo - http://es.wikipedia.org/wiki/Token)

Entonces, la fórmula propuesta es:

$$CFPI_i = TC_i * \log_2(TC_i)$$

Conociendo el tamaño corregido de cada primitiva funcional, se suma dichos valores para saber cuál es el **tamaño funcional total** del sistema. Al valor obtenido se le llama CFP (Primitiva Funcional Corregida) y representa el tamaño total corregido de todas las primitivas funcionales del sistema:

$$CFP = \sum_i CFP_i$$

Hay casos en los que puede haber mucha variación de la complejidad de las primitivas funcionales de nuestro sistema, es decir, que unas sean más complejas y más costosas de implementar que otras, lo cual es muy habitual. En estos casos, DeMarco propone clasificar las primitivas funcionales en diferentes categorías y asignarle a cada una de ellas un *peso* o factor de corrección en función de la complejidad que presenten.

Tabla 3.12 Clasificación de las primitivas funcionales

Categoría	Peso
Separación	0.6
Unión	0.6
Encauzado de datos	0.3
Actualización simple	0.5
Almacenamiento	1.0
Edición	0.8
Verificación	1.0
Manipulación de texto	1.0

Tabla 3.13 Categoría y factores de corrección, según la complejidad de las primitivas funcionales

Categoría	Peso
Sincronización	1.5
Generación de salidas	1.0
Muestreo	1.8
Análisis tabular	1.0
Aritmética	0.7
Iniciación	1.0
Cálculo matemático	2.0
Manipulación de dispositivos	2.5

B. Para sistemas orientados a los datos

Los sistemas orientados a los datos implementan o gestionan una base de datos. Por lo tanto, la mayoría del esfuerzo de desarrollo para estos sistemas está destinado a la implementación de la propia base de datos. El indicador principal que se utilice para la medida del tamaño funcional sea el factor OB, es decir, el número de **objetos** del modelo de datos del sistema (habitualmente un modelo E/R, en el que 'objeto' equivaldría a 'entidad').

Es necesario un factor de corrección que contemple el hecho de que unos objetos son más costosos de implementar que otros. DeMarco propone los siguientes factores de corrección o pesos, en función de la interrelación de esos objetos con otros (factor REi, o número de **interrelaciones** del modelo de datos del sistema en las que está involucrado el objeto i). A estos factores de corrección se les llama COBI (Corrected Object Increment), y deberán ser asignados a todos los objetos (entidades) del modelo de datos del sistema (Tabla 3.14).

Tabla 3.14 Factores de corrección de (pesos) para los objetos de un sistema, en función de sus interrelaciones con otros.

RE_i	COBI
1	1.0
2	2.3
3	4.0
4	5.8
5	7.8
6	9.8

Los valores COBI no son precisos, puesto que dependen del entorno de desarrollo y de las herramientas que se utilicen para implementar la base de datos.

El valor resultante para la métrica Bang, será entonces la suma de los valores COBI de cada objeto (entidad) del sistema, es decir:

$$COB = \sum_i COBI_i$$

COB (Corrected Object) es el **tamaño funcional total** de la aplicación medida.

C. Para sistemas híbridos

En el caso de sistemas híbridos, DeMarco propone calcular independientemente las dos métricas Bang comentadas en los apartados A y B anteriores. Es decir, se trata de:

- Considerar el sistema híbrido como un sistema orientado a la función, para calcular así la primera métrica Bang.
- Considerar el sistema híbrido como un sistema orientado a los datos, para calcular así la segunda métrica Bang.

De esta forma se tiene dos medidas del tamaño funcional, una que representa el esfuerzo de desarrollo de la parte funcional, y otra que representa el esfuerzo de desarrollo de la parte orientada a los datos.

3.2.8.3 Técnica Tabla Objeto-Evento (OET)

La técnica de Tabla Objeto-Evento, *"es una técnica útil de modelamiento de dominio orientado a objetos, que se basa en la interacción entre objetos y eventos, permitiendo elaborar mejor los modelos conceptuales y verificar las actividades definidas en la especificación de requisitos"*[1].

Para el desarrollo del modelo conceptual orientado a objetos, se requiere definir todos los tipos de objetos existentes y su comportamiento. Esta técnica está basada en el concepto de paso de mensajes como interacción entre objetos, y se propone una alternativa para la comunicación y participación conjunta de eventos de negocio.

Como ejemplo, en una biblioteca podemos identificar dos tipos de objetos de dominio: Miembro y Copia (*Member* and *Copy)*. Un tipo de evento en este dominio es el pedir prestado una copia (*borrowing* de un *copy)*. Este evento afecta a ambos tipos de objetos, ya que modifica el estado de *copy* y el estado de *member*. Cuando se usa el paso de mensajes como interacción entre objetos, existen dos escenarios posibles: o bien *member* envía un mensaje a *copy,* o *copy* envía un mensaje a *member*.

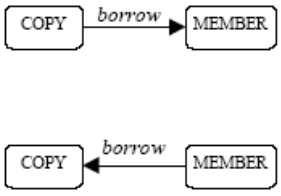


Figura 3.6 Dos posibles escenarios para el préstamo de una copia

Además se tiene el tipo de objeto de dominio *LOAN*; luego de *borrow-event* se creará el objeto préstamo (*loan).* En este caso, tres objetos actúan simultáneamente con un solo evento y deben ser notificados de acuerdo a la ocurrencia de *borrow-event.* Cuando se realiza el paso de mensajes, esto lleva a 9 escenarios de interacción posible. Si 4 objetos tienen sincronismo en la ocurrencia de un evento, tendremos 48 posibles escenarios en el paso de mensajes, considerando algunos escenarios más importantes y adecuados que otros.

[1] (Documento "The object-event table: A useful technique in object-oriented domain modeling", por M. Snoeck, pág. 2.)

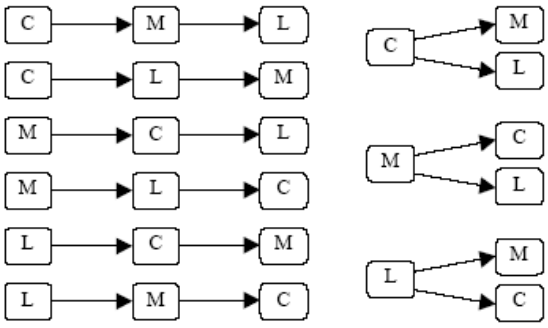


Figura 3.7 Posibles escenarios cuando tres objetos son envueltos en un solo evento

La alternativa que se propone es un modelo basado en la interacción de objetos que son afectados por eventos. Como los objetos deben ser notificados por la ocurrencia de los eventos, depende de la implementación. Cuando se requiere hacer un modelo donde los objetos están involucrados con los tipos de eventos, se puede usar una técnica simple que es la *técnica de tabla objeto-evento.*

En la siguiente tabla se muestra la posible tabla objeto-evento de una biblioteca como ejemplo. La tabla muestra un evento llamado *cr_member* que afecta solo al objeto *member,* la adquisición de una copia *acquire* solo afecta a *copy,* pero que *borrowing* y *return* de una copia afecta a *member,* a *copy* y al objeto *loan.*

Tabla 3.15 Tabla objeto-evento para una biblioteca

	MEMBER	COPY	LOAN
cr_member	×		
acquire		×	
borrow	×	×	×
return	×	×	×
...			

Como explicación de la tabla anterior, podemos saber los tipos de objetos que son afectados por otros tipos de eventos. Esta es una forma de conocer y verificar los aspectos internos importantes del sistema, la consistencia y coherencia en la especificación de requisitos.

Eventos de Negocio

Definición

Dentro del modelamiento de dominio se utilizan los eventos de negocio como parte del diseño de la tabla objeto-evento; por esta razón, no se toma en cuenta los eventos de información del sistema como son las acciones del teclado o ratón.

Para considerarse eventos de negocio debe satisfacer las siguientes propiedades:

- Un evento de negocio ocurre o es reconocido dentro de un período de tiempo. Ejemplo: el evento *pago* se incluye dentro de la lista de los tipos de eventos relevantes, esto significa que dentro de cualquier actividad que se efectúe en un período de tiempo se realizará un pago.
- Un evento de negocio marca algo real que ha ocurrido, sin considerar los eventos de información del sistema.
- Un evento de negocio no se descompone, es decir no se divide en subeventos.

Los eventos de negocio deben ser asignados de preferencia con el infinitivo de un verbo en lugar de un verbo conjugado. Por ejemplo: *pagar*, pero no *paga*.

Tabla Objeto-Evento: una técnica simple de modelamiento

El modelo de la empresa no solo esta compuesta por los objetos de la empresa, sino también de los objetos de negocio. Supongamos que, para la biblioteca ejemplo, los tipos de objetos de dominio son *MEMBER, COPY* y *LOAN*. Como los eventos de negocio relevantes identificamos *enter* (se añade un nuevo miembro a la biblioteca), *leave* (un miembro deja la biblioteca), *acquire* (una nueva copia se adquiere), *classify* (una copia está clasificada), *borrow* (una copia es prestada por un miembro), *renew* (un crédito se renueva), *return* (una copia se regresa a la biblioteca), *remove_copy* (una copia se elimina de la biblioteca), y *lose* (una copia se pierde en lugar de ser devuelta).

Para cada uno de los eventos de negocio, podemos identificar los objetos de dominio que participan en cada uno de los eventos. Por ejemplo, cuando una copia se pide prestado *(borrow)*, hay un ejemplar *(copy)* que se involucra, un miembro *(member)* y un

préstamo *(loan)*. El evento *borrow* va a ser modificado en el estado de *copy*, modificado en el estado de *member* y creado en *loan*. Esto puede ser fácilmente documentado en la tabla objeto-evento, poniendo ´C´, ´M´ y ´E´, respectivamente creación, modificación y terminación de un objeto.

Tabla 3.16 OET para el ejemplo biblioteca

	MEMBER	COPY	LOAN
enter	C		
leave	E		
acquire		C	
classify		M	
borrow	M	M	C
renew	M	M	M
return	M	M	E
remove_copy		E	
lose	M	E	E

Definición formal de la Tabla Objeto-Evento

El conjunto **A** es el universo de los tipos de eventos y **O** es el universo de los tipos de objetos de dominio para el dominio especificado. Entonces en cada fila de la tabla objeto-evento se le identifica con cada evento de negocio y cada columna es identificada con el tipo de objeto, los cuales pueden ser definidos por lo siguiente:

$\mathcal{T} \subseteq O \times A \times \{' \ ' , C, M, E\}$ such that
$\forall P \in O, \forall a \in A : (P,a,' \ ' \in) \mathcal{T}$ or $(P,a,C) \in \mathcal{T}$ or $(P,a,M) \in \mathcal{T}$ or $(P,a,E) \in \mathcal{T}$

Definición formal de la Tabla objeto-evento[1]

Cada celda de la tabla indica si un tipo de objeto participa en un tipo de evento o no. De esta manera, un subconjunto de los tipos de eventos se asigna a cada tipo de objeto. Este subgrupo se llama **el alfabeto de un objeto** tipo P y se denota SAP. Contiene todos los tipos de eventos que son relevantes para el tipo de objeto particular. Además, el alfabeto de P es particionado en tres conjuntos disjuntos c(P), m(P) y e(P) con:

c(P) = {The set of event types that create an occurrence of type P} $\subseteq \in S_A P$
m(P) = {The set of event types that modify an occurrence of type P} $\subseteq S_A P$
e(P)= {The set of event types that end an occurrence of type P} $\subseteq S_A P$

[1] Documento "The object-event table: A useful technique in object-oriented domain modeling", por M. Snoeck, pág.6.

Formally

$\forall P \in O: c(P) = \{a \in A \mid (P,a,' C') \in T\}$

$m(p) = \{a \in A \mid (P,a,' M') \in T\}$

$e(P) = \{a \in A \mid (P,a,' E') \in T\}$

$S_A P = c(P) \cup m(P) \cup e(P)$

Fundamento formal de los conjuntos de eventos[1]

Para el ejemplo biblioteca dado, significa que:

A ={enter, leave, acquire, classify, borrow, renew, return, remove_copy, lose}
O = {MEMBER, COPY, LOAN}

The partitions of the alphabets of the object types COPY, MEMBER and LOAN are as follows:

S_ACOPY = {acquire, classify, borrow, renew, return, remove_copy, lose} with
c(COPY) = {acquire}
m(COPY) = {classify, borrow, renew, return}
e(COPY) = {remove_copy, lose}

S_AMEMBER = {enter, borrow, renew, return, lose, leave} with
c(MEMBER) = {enter}
m(MEMBER) = {borrow, renew, return, lose}
e(MEMBER) = {leave}

S_ALOAN = {borrow, renew, return, lose} with
c(LOAN) = {borrow}
m(LOAN) = {renew}
e(LOAN) = {return, lose}

Definición del Objeto

Para definir las reglas de control en la consistencia de los objetos, se debe determinar el comportamiento de cada uno de ellos a través de escenarios[2]. Por ejemplo *borrow^renew^return.* Si los operadores de iteración se utilizan en el ciclo de vida de los objetos, el número de escenarios válidos para un tipo de objeto es infinito. Supongamos, por ejemplo, que el ciclo de vida de una especificación del objeto *LOAN* es el siguiente:

$$LOAN = borrow.(renew^*).(return + lose)$$

Es decir, un préstamo se crea cuando una copia se toma prestada. El préstamo puede ser renovado un número arbitrario de veces. Por último, el préstamo llega a su fin cuando el ejemplar se devuelve a la biblioteca o cuando se haya registrado como perdido.

[1] (Documento "The object-event table: A useful technique in object-oriented domain modeling", por M. Snoeck, pág.6)
[2] (Un escenario es la secuencia de tipos de eventos - http://es.wikipedia.org/wiki/Escenario)

El conjunto de escenarios válidos para *LOAN* es:

{borrow^return, borrow^lose, borrow^renew^return, borrow^renew^lose,
borrow^renew^renew^return, borrow^renew^renew^lose, …}

El conjunto de escenarios válidos definidos por un tipo de objeto se llama **lengua de tipo de objeto.**

3.2.8.4 Medición de los Modelos Conceptuales basado en eventos y orientado a objetos

El Modelo Conceptual se utiliza para modelar, estructurar y analizar un dominio o parte de él. La definición de un modelo de dominio es parte de los requisitos de ingeniería en el desarrollo de un sistema de software. Todas las reglas descritas en el modelo de dominio deben ser soportadas por el sistema. El análisis orientado a objetos pretende realizar el análisis del modelado y de los requisitos del sistema (interfaz de usuario, almacenamiento de datos, flujo de datos, calidad en los requisitos, etc.).

Tabla Objeto-Evento (OET)

El conjunto A es usado para denotar el universo de tipos de eventos relevantes. Se presenta un ejemplo donde se presenta el préstamo *(loan)* de una biblioteca. Se asume que el modelo conceptual de la biblioteca es inicializada por el siguiente universo de tipos de eventos:

A = {*start_membership, end_membership, acquire, catalogue, borrow, renew, return, sell, reserve, cancel, fetch, lose*}

Características de los objetos

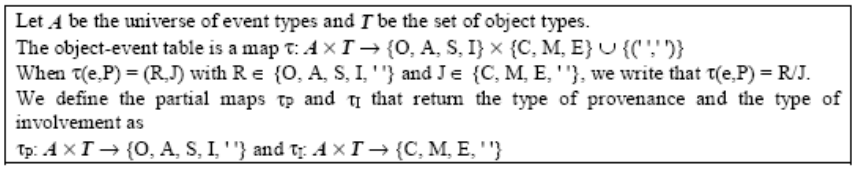
Los objetos están caracterizados por lo siguiente:

- Cada objeto en el modelo conceptual forma parte del mundo real del sistema.
- Los objetos son descritos por el número de propiedades. Las propiedades son especificadas en un tipo de objeto (Ejemplo: un clasificador de UML con un <<object type>> stereotype).
- Los objetos existen por un cierto período de tiempo.

- Los objetos siempre participan en dos eventos reales: la creación de eventos y la finalización de eventos. La participación en la finalización de eventos no significa que el objeto es destruido, sino que el objeto no puede seguir participando de un evento real.

La participación de eventos son modelados por el OET. Los tipos de participación de eventos son: *create (C), modify. (M),* o *end (E).* Una modificación de un tipo de evento para un tipo de objeto no es la creación de una instancia de un objeto, ni tampoco la terminación de su existencia. La modificación de un evento es el cambio de estado de un objeto. Las operaciones de los tipos de eventos en el alfabeto de los tipos de objetos son: *acquired through propagation (A)* (adquirido por propagación), *inherited (I)* (heredado), *inheriited in a specialised version (S)* (heredado en una versión especializada), *the class of own event types (O)* (clase de los propios tipos de eventos).

A continuación se especifica el fundamento básico sobre la tabla objeto-evento, donde el conjunto de tipos de objetos relevantes son el universo de tipos de eventos de A denotados por el capital T.


Let $\mathcal{A}$ be the universe of event types and $\mathcal{T}$ be the set of object types.
The object-event table is a map $\tau: \mathcal{A} \times \mathcal{T} \rightarrow \{O, A, S, I\} \times \{C, M, E\} \cup \{(' ',' ')\}$
When $\tau(e,P) = (R,J)$ with $R \in \{O, A, S, I, ' '\}$ and $J \in \{C, M, E, ' '\}$, we write that $\tau(e,P) = R/J$.
We define the partial maps τ_P and τ_I that return the type of provenance and the type of involvement as
$\tau_P: \mathcal{A} \times \mathcal{T} \rightarrow \{O, A, S, I, ' '\}$ and $\tau_I: \mathcal{A} \times \mathcal{T} \rightarrow \{C, M, E, ' '\}$

Fundamento Tabla objeto-evento[1]

[1] (Documento On the Measurement of Event-Based Object-Oriented Conceptual Models, Geert Poels, pág. 3)

Tabla 3.17 OET de Biblioteca[1]

	ITEM	VOLUME	COPY	RESERVATION	MEMBER	LOAN	NOT_RENEWABLE_LOAN	RENEWABLE_LOAN
acquire	O/C							
acquire_volume		S/C						
acquire_copy			S/C					
catalogue	O/M	I/M	I/M					
sell	O/E							
sell_volume		S/E						
sell_copy			S/E					
reserve			A/M	O/C	A/M			
cancel			A/M	O/E	A/M			
fetch			A/M	O/E	A/M			O/C
start_membership					O/C			
end_membership					O/E			
borrow					A/M	O/C		
create_not_renewable_loan		A/M			A/M		S/C	
create_renewable_loan			A/M		A/M			S/C
return		A/M	A/M		A/M	O/E	I/E	I/E
lose					A/M	O/E		
lose_volume		A/E			A/M		S/E	
lose_copy			A/E		A/M			S/E
renew			A/M		A/M			O/M

La siguiente figura muestra el modelo estructural de biblioteca. Note que los tipos de objetos *ITEM* tiene dos subtipos: *VOLUME* y *COPY*. Asumiendo que *volumes* puede ser *borrowed* (prestados), pero *loans* (préstamos) no pueden ser *renewed* (renovados), entonces el tipo de objeto *LOAN* debe ser especificado.

[1] (Documento On the Measurement of Event-Based Object-Oriented Conceptual Models, Geert Poels, pág. 4)

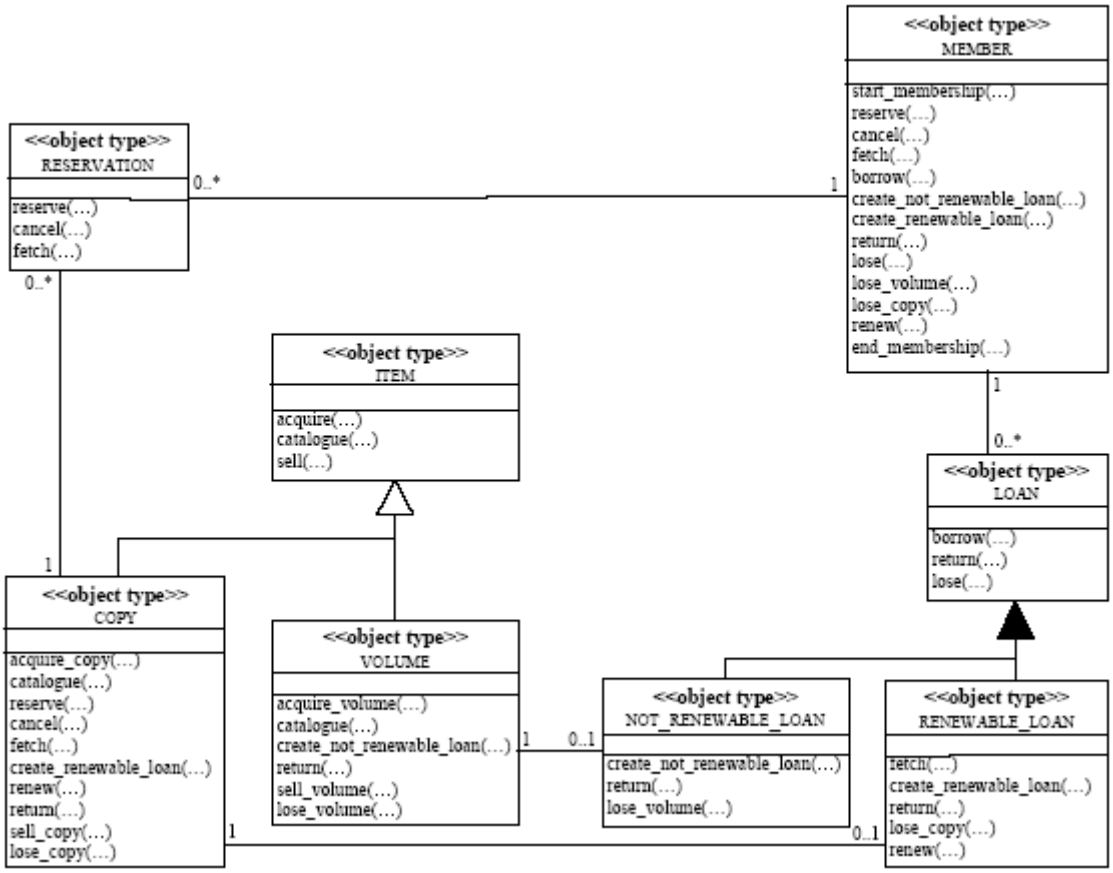


Figura 3.8 Modelo Estructural Biblioteca

OET basado en la medición

Para las definiciones de las mediciones, se debe asumir un modelo conceptual ***S*** con un universo de tipo de eventos ***A***, un conjunto de objetos ***T*** relevante para ***A***, y una tabla objeto-evento ***t.*** Se usa el símbolo **#** para la cardinalidad de los conjuntos.

Medición del Tamaño: El tamaño de un proyecto de software se determina de acuerdo al número de elementos que se usarán para definir, especificar, construir y componer. El tamaño puede ser expresado y medido por medio de la participación de eventos. Entre más tipos de eventos y más tipos de objetos se vean involucrados, mayores serán las operaciones posibles a implementar en la definición de clases. Por lo tanto, las participaciones de eventos proporcionan una temprana estimación del tamaño de las clases, siendo útil para la determinación de costos que tenga el proyecto. Las participaciones de eventos son la base para las estimaciones de gastos y esfuerzo, fijación de precios y toma de decisiones.

Tabla 3.18 Tamaños basados en OET[1]

MEASUREMENT OBJECT	MEASURE DESCRIPTION	MEASURE DEFINITION
Object type: $P \in T$	Count of Event Participations	$CEP(P) = \#\{e \in A \mid \tau(e,P) \neq \text{' '/' '}\}$
Conceptual model: S	Level of Object-Event Interaction	$LOEI(S) = \sum_{P \in T} CEP(P)$

3.2.8.5 Técnica de Minería de Datos

Cabena et al. (1998) dice que *"para obtener la calidad de los productos de software en tiempos de entrega y costos es importante realizar el análisis adecuado en las primeras etapas del ciclo de vida de proyectos. Los errores ocasionados en dichas etapas producen dificultades en el mantenimiento, baja reutilización y comportamiento defectuoso de los programas"*[2].

Estas son las principales causas por las que las ERS[3] está adquiriendo cada vez mayor importancia, debido a la necesidad de obtener datos objetivos que contribuyan a mejorar la calidad desde las primeras etapas del proyecto.

El fundamento para la medición de la especificación de requisitos es el desarrollo de métricas, que ayudan en la determinación del tamaño y funcionalidad del software. Entre las métricas más utilizadas se encuentran las de puntos de función, métricas Bang o los puntos objeto.

Las técnicas de minería de datos son herramientas útiles para la explotación profunda y extracción de información de los proyectos de software.

Técnicas de Minería de Datos

La minería de datos sustituye el análisis basado en la verificación de datos por un enfoque dirigido al descubrimiento de información sin necesidad de formular previamente una hipótesis.

[1] (Documento On the Measurement of Event-Based Object-Oriented Conceptual Models, Geert Poels, pág. 6)

[2] (Cabena, P., Hadjinian, P., Stadler, R., Verhees, J. Y Zanasi, A. "*Discovering Data Mining. From Concept to Implementation*", Prentice Hall, 1998.)

[3] (ERS, Especificación de Requisitos del Sistema)

Clasificación de las Técnicas de Minería de Datos

Los algoritmos de minería de datos se clasifican en dos grandes categorías:

1. Supervisados o predictivos
2. No supervisados o de descubrimiento del conocimiento.

Los algoritmos **supervisados o predictivos** predicen el valor de un atributo (*etiqueta*) de un conjunto de datos. A partir de datos cuya etiqueta se conoce se induce una relación entre dicha etiqueta y otra serie de atributos. Esta forma de trabajar se conoce como *aprendizaje supervisado* y se desarrolla en dos fases: entrenamiento (construcción de un modelo usando un subconjunto de datos con etiqueta conocida) y prueba (prueba del modelo sobre el resto de los datos).

Si en una aplicación no se puede determinar una solución predictiva, se debe recurrir a los métodos **no supervisados o de descubrimiento del conocimiento** que descubren patrones y tendencias en los datos actuales (no utilizan datos históricos).

En la tabla siguiente se muestran algunas de las técnicas de minería de ambas categorías.

Tabla 3.19 Clasificación de las técnicas de Minería de Datos[1]

Supervisados	**No supervisados**
Árboles de decisión	Detección de desviaciones
Inducción neuronal	Segmentación
Regresión	Agrupamiento ("clustering")
Series temporales	Reglas de asociación
	Patrones secuenciales

[1] (REVISTA COLOMBIANA DE COMPUTACIÓN, "Obtención y Validación de Modelos de Estimación de Software Mediante Técnicas de Minería de Datos", Volumen 3, número 1, Pág. 55)

Etapas de la Minería de Datos

1. Determinación de objetivos

2. Preparación de datos

- Selección: Identificación de las fuentes de información externas e internas y selección del subconjunto de datos necesario.
- Preprocesamiento: estudio de la calidad de los datos y determinación de las operaciones de minería que se pueden realizar.

3. Transformación de datos: conversión de datos en un modelo analítico.

4. Minería de datos: tratamiento automatizado de los datos seleccionados con una combinación apropiada de algoritmos.

5. Análisis de resultados: interpretación de los resultados obtenidos en la etapa anterior, generalmente con la ayuda de una técnica de visualización.

6. Asimilación de conocimiento:

Las etapas de la minería de datos se realizan en un orden, pero el proceso puede ser iterativo, existiendo retroalimentación entre los mismos. No todos los pasos requieren el mismo esfuerzo, las etapas de preprocesamiento es la más costosa porque representa aproximadamente el 60% del esfuerzo total, mientras que la etapa de minería representa el 10%.

6.3.- Aplicaciones de la minería de datos en la medición del software

Las técnicas de minería de datos se están utilizando desde hace varios años para la obtención de patrones en los datos y para la extracción de información valiosa en el campo de la Ingeniería del Software.

Entre estas aplicaciones podemos citar:

- Utilización de árboles de decisión en la construcción de modelos para el desarrollo de software.
- Aplicación de técnicas de *clustering* en la planificación del mantenimiento y en la estimación de la fiabilidad del software.
- Uso de redes neuronales en la predicción de riesgos de mantenimiento en módulos de programa.

La mayor parte de los trabajos realizados están dirigidos a la obtención de modelos de estimación de esfuerzo de desarrollo y modelos de predicción de diferentes aspectos de la calidad del software. Las métricas utilizadas constituyen la base para la construcción de los modelos y posterior validación de los mismos.

6.3.1.- Ejemplo de aplicación

A continuación se muestra un ejemplo de aplicación de la técnica mencionada en la construcción de modelos de estimación para determinar el tamaño de un proyecto.

6.3.1.1.- Estudio previo de los datos utilizados

Se dispone de datos referentes a 42 proyectos implementados con Informix-4GL[1]. Los sistemas desarrollados son aplicaciones de contabilidad con las características de sistemas comerciales, cada uno incluye alguno de los siguientes subsistemas: Compras, ventas, inventario, finanzas y ciclos de producción.

Toda la información de los proyectos se ha dividido en dos grupos:

1. **Primer grupo:** descomposición en módulos o componentes para obtención de atributos de cada módulo. Contiene 1537 registros.
2. **Segundo grupo:** datos globales de cada uno de los proyectos. Contiene 42 registros.

[1] (Lenguaje de cuarta generación (Informix-4GL))

6.3.1.2.- Descripción de los atributos de los módulos:

TYPECOMP: Tipo de componente (1: menú, 2: entrada, 3: informe/consulta)
OPTMENU: Número de opciones (sólo para componentes de tipo 1)
DATAELEMEN: Número de elementos de datos (sólo para componentes de tipo 2 y 3)
RELATION: Número de relaciones (sólo para componentes de tipo 2 y 3)
LOC: Número de líneas de código del módulo.

A continuación se realiza un estudio estadístico de cada uno de los atributos de los módulos que componen el proyecto. Se puede observar en todos los casos que el tamaño de cada uno de los módulos no es muy grande, siendo los más pequeños los componentes de tipo menú. En el caso de otros componentes el número de elementos de datos y relaciones no es muy alto, por esta razón los valores altos de los atributos tienen suficiente peso para influir en las decisiones y conclusiones del proyecto.

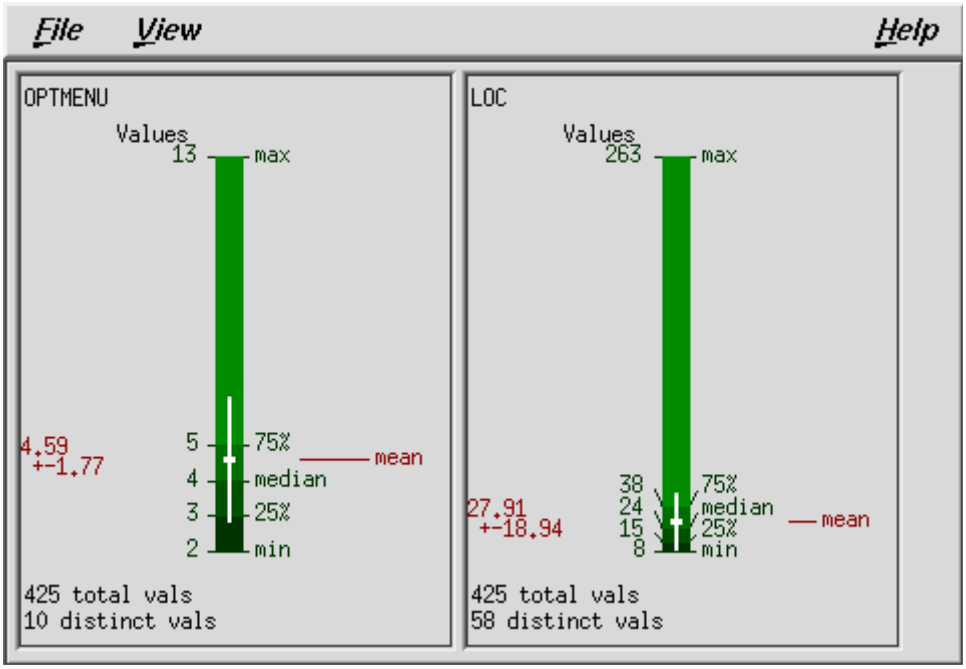


Figura 3.9 Estadísticas para componentes de tipo 1 (menú)

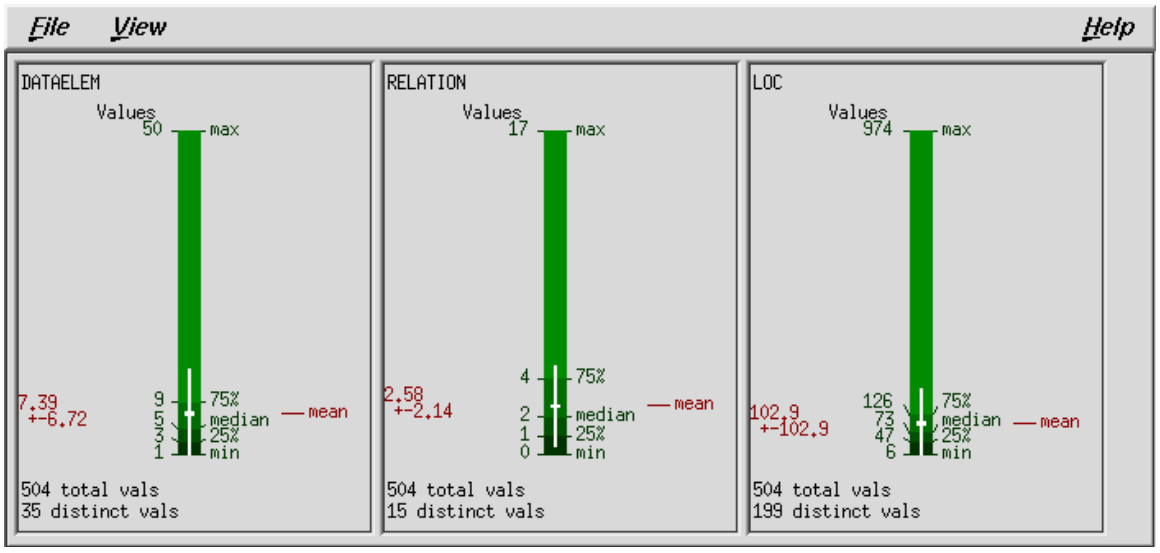


Figura 3.10 Estadísticas para componentes de tipo 2 (entradas)

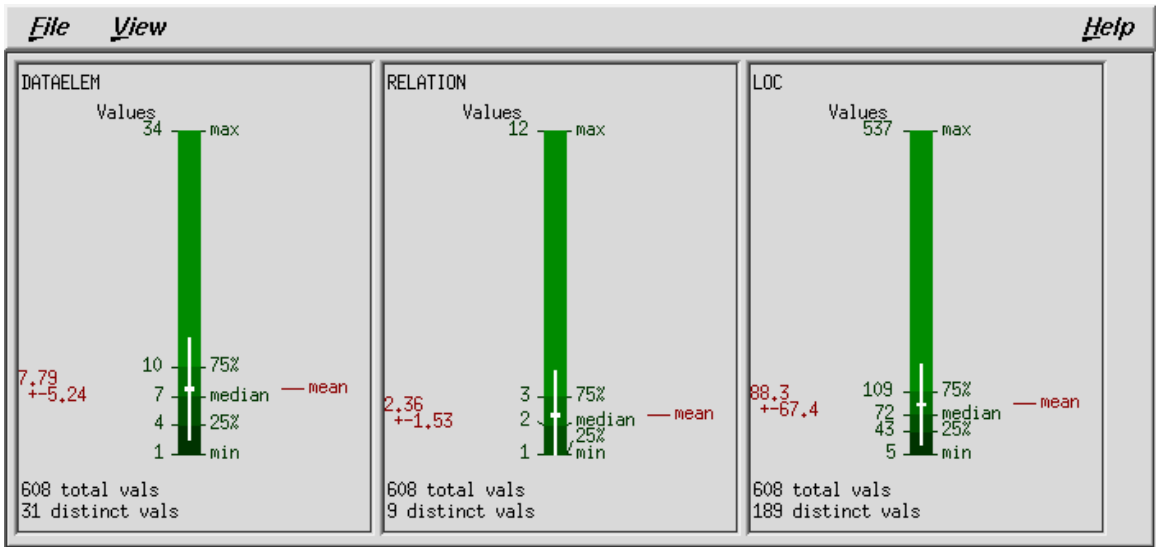


Figura 3.11 Estadísticas para componentes de tipo 3 (informes/consultas)

6.3.1.3.- Descripción de los atributos de proyectos

A continuación se mencionan los atributos más utilizados en el método Mark II para el cálculo de los puntos de función. El método considera que cada sistema está formado por transacciones lógicas[1].

[1] (Una transacción lógica es una única combinación entrada/proceso/salida activada por un único evento que tiene significado para el usuario o es el resultado de una consulta o extracción de información - http://es.wikipedia.org/wiki/Transaccion_Logica)

LOC: Número de líneas de código
NOC: Número de componentes
NTRNSMKII: Número de transacciones
MKIIINPTMKII: Número total de entradas
ENTMKII: Número de entidades referenciadas
OUTMKII: Número total de salidas (elementos de datos sobre todas las transacciones)
UFPMKII: Número de puntos de función no ajustados MKII

Para contabilizar las entidades (ficheros lógicos), se dividen en:

- **Primarias**: aquellas para las que se construye el sistema con la intención de recolectar y almacenar datos).
- **No primarias**: las utilizadas como referencia, validación, etc.
- **Sistema**: todas las entidades no primarias se engloban en una única.

Se contabiliza el número de entidades a las que se hace referencia en cada transacción, no el número de referencias. En cuanto al número de entradas y salidas, se cuentan los tipos de elementos de datos, no ocurrencias (aplicable también a las tablas). No se cuentan elementos de datos de cabeceras o pies de pantalla no generados específicamente para esa pantalla, tampoco se contabilizan etiquetas, cajas, etc. Los mensajes de error se consideran ocurrencias del tipo de dato "mensaje de error"(análogo para los mensajes de operador).

A continuación se muestra el estudio estadístico, y se determina que el tamaño de los proyectos estudiados se encuentra entre 726 y 8.888 líneas de código.

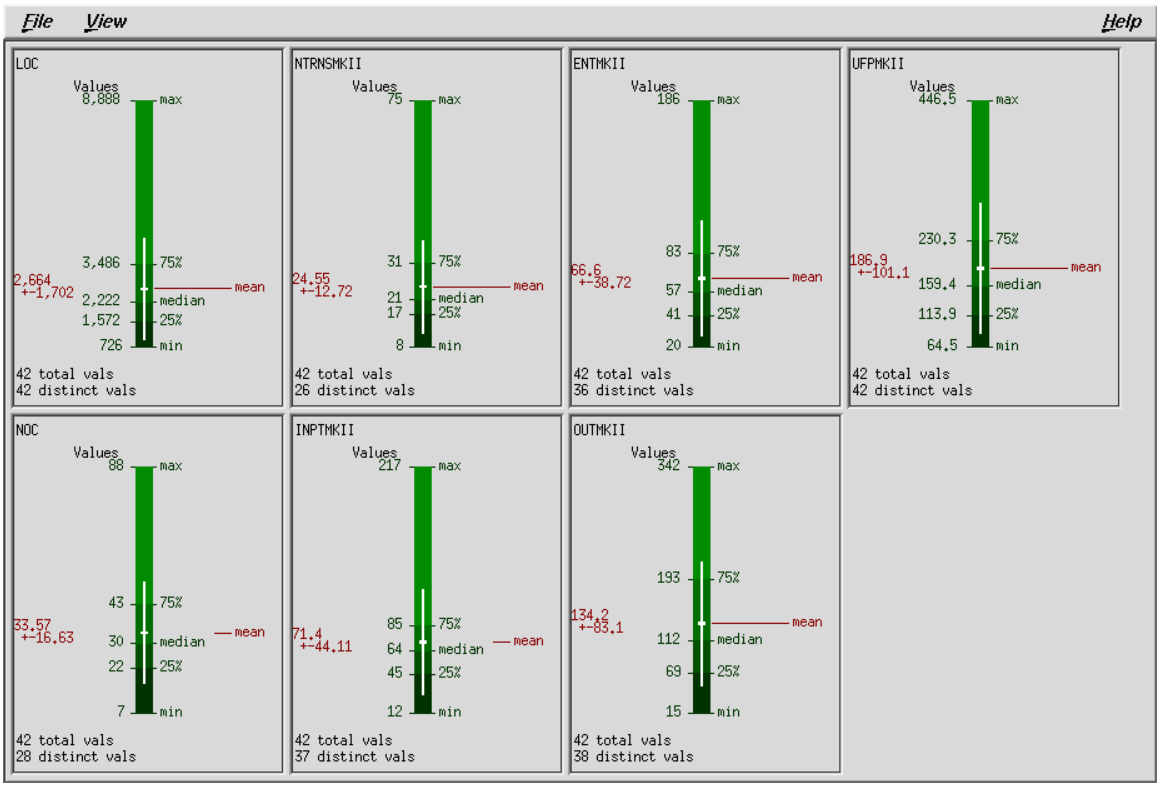


Figura 3.12 Estadísticas para atributos del proyecto

3.2.8.6 Técnica de Estimación Puntos de Casos de Uso

La estimación mediante el análisis de Puntos de Casos de Uso es un método propuesto originalmente por Gustav Karner de Objectory AB (1993), *"consiste en la estimación del tiempo de desarrollo de un proyecto mediante la asignación de "pesos" a un cierto número de factores que lo afectan, para finalmente, calcular el tiempo total estimado para el proyecto a partir de esos factores"*[1].

La estimación por Puntos de Caso de Uso resulta muy efectiva para estimar el esfuerzo requerido en el desarrollo de los primeros Casos de Uso de un sistema, si se sigue una aproximación iterativa como el Proceso Unificado de Rational (RUP).

Para la especificación de Casos de Uso se realiza la descripción del Escenario principal que relata las acciones del actor y las del sistema durante una utilización típica, y un

[1] (Fue introducido por Gustav Karner en su tesis en 1993 (Universidad de Linkoping) y supervisado por Ivar Jacobson en Objectory AB. Ha sido analizado posteriormente en otros estudios, como la tesis de Kirsten Ribu (Univerdidad de Oslo) en 2001)

conjunto de Escenarios alternativos que relatan las condiciones de excepción, o las alternativas de llevar a cabo la secuencia de sucesos.

Cálculo de Puntos de Casos de Uso sin ajustar

El primer paso para la estimación consiste en el cálculo de los Puntos de Casos de Uso sin ajustar. Este valor, se calcula a partir de la siguiente ecuación:

$$\mathbf{UUCP = UAW + UUCW}$$

Donde:

UUCP: Puntos de Casos de Uso sin ajustar

UAW: Factor de Peso de los Actores sin ajustar

UUCW: Factor de Peso de los Casos de Uso sin ajustar

Factor de Peso de los Actores sin ajustar (UAW)

Este valor se calcula mediante un análisis de la cantidad de Actores presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los Actores se establece teniendo en cuenta en primer lugar si se trata de una persona o de otro sistema, y en segundo lugar, la forma en la que el actor interactúa con el sistema. Los criterios se muestran en la siguiente tabla:

Tabla 3.20 Factores de Peso de los Actores

Tipo de Actor	Descripción	Factor de Peso
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (API, Application Programming Interface)	1
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto	2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica	3

Factor de Peso de los Casos de Uso sin ajustar (UUCW)

Este valor se calcula mediante un análisis de la cantidad de Casos de Uso presentes en el sistema y la complejidad de cada uno de ellos. *"La complejidad de los Casos de Uso se establece teniendo en cuenta la cantidad de transacciones efectuadas en el mismo, donde una transacción se entiende como una secuencia de actividades completa o no se efectúa ninguna de las actividades de la secuencia"* [1]. Los criterios se muestran en la siguiente tabla:

Tabla 3.21 Factores de Peso de los Casos de Uso

Tipo de Caso de Uso	Descripción	Factor de Peso
Simple	El Caso de Uso contiene de 1 a 3 transacciones	5
Medio	El Caso de Uso contiene de 4 a 7 transacciones	10
Complejo	El Caso de Uso contiene más de 8 transacciones	15

Cálculo de Puntos de Casos de Uso ajustados

Una vez que se tienen los Puntos de Casos de Uso sin ajustar, se debe ajustar éste valor mediante la siguiente ecuación:

$$\mathbf{UCP = UUCP \times TCF \times EF}$$

Donde:

UCP: Puntos de Casos de Uso ajustados

UUCP: Puntos de Casos de Uso sin ajustar

TCF: Factor de complejidad técnica

EF: Factor de ambiente

[1] (Documento "ESTIMACIÓN DEL ESFUERZO BASADA EN CASOS DE USO", por Mario Peralta, http://www.itba.edu.ar/capis/webcapis/planma.html)

Factor de complejidad técnica (TCF)

Este coeficiente se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema. Cada uno de los factores se cuantifica con un valor de 0 a 5, donde 0 significa un aporte irrelevante y 5 un aporte muy importante. En la tabla 3.22 se muestra el significado y el peso de cada uno de éstos factores:

Tabla 3.22 Factores de Complejidad Técnica

Factor	Descripción	Peso
T1	Sistema distribuido	2
T2	Objetivos de performance o tiempo de respuesta	1
T3	Eficiencia del usuario final	1
T4	Procesamiento interno complejo	1
T5	El código debe ser reutilizable	1
T6	Facilidad de instalación	0.5
T7	Facilidad de uso	0.5
T8	Portabilidad	2
T9	Facilidad de cambio	1
T10	Concurrencia	1
T11	Incluye objetivos especiales de seguridad	1
T12	Provee acceso directo a terceras partes	1
T13	Se requieren facilidades especiales de entrenamiento a usuarios	1

El Factor de complejidad técnica se calcula mediante la siguiente ecuación:

$$TCF = 0.6 + 0.01 \times \Sigma (Peso_i \times Valor\ asignado_i)$$

Factor de ambiente (EF)

Las habilidades y el entrenamiento del grupo involucrado en el desarrollo tienen un gran impacto en las estimaciones de tiempo. Estos factores son los que se contemplan en el cálculo del Factor de ambiente. El cálculo del mismo es similar al cálculo del Factor de complejidad técnica, es decir, se trata de un conjunto de factores que se cuantifican con valores de 0 a 5.

En la tabla 3.23 se muestra el significado y el peso de cada uno de éstos factores.

Tabla 3.23 Factor de Ambiente

Factor	Descripción	Peso
E1	Familiaridad con el modelo de proyecto utilizado	1.5
E2	Experiencia en la aplicación	0.5
E3	Experiencia en orientación a objetos	1
E4	Capacidad del analista líder	0.5
E5	Motivación	1
E6	Estabilidad de los requerimientos	2
E7	Personal part-time	-1
E8	Dificultad del lenguaje de programación	-1

- Para los factores E1 al E4, un valor asignado de 0 significa sin experiencia, 3 experiencia media y 5 amplia experiencia (experto).
- Para el factor E5, 0 significa sin motivación para el proyecto, 3 motivación media y 5 alta motivación.
- Para el factor E6, 0 significa requerimientos extremadamente inestables, 3 estabilidad media y 5 requerimientos estables sin posibilidad de cambios.
- Para el factor E7, 0 significa que no hay personal part-time (es decir todos son full-time), 3 significa mitad y mitad, y 5 significa que todo el personal es part-time (nadie es full-time).
- Para el factor E8, 0 significa que el lenguaje de programación es fácil de usar, 3 medio y 5 que el lenguaje es extremadamente difícil.

El Factor de ambiente se calcula mediante la siguiente ecuación:

$$EF = 1.4 - 0.03 \times \Sigma (Peso_i \times Valor\ asignado_i)$$

Estimación del esfuerzo basado en los Puntos de Casos de Uso

Karner (1993) originalmente sugirió "*que para la estimación del esfuerzo cada Punto de Casos de Uso requiere **20 horas-hombre**"*[1]**.** Posteriormente, surgieron otros criterios:

- Se contabilizan cuántos factores de los que afectan al Factor de ambiente están por debajo del valor medio (3), para los factores E1 a E6.
- Se contabilizan cuántos factores de los que afectan al Factor de ambiente están por encima del valor medio (3), para los factores E7 y E8.
- Si el total es 2 o menos, se utiliza el factor de conversión 20 horas-hombre/Punto de Casos de Uso, es decir, un Punto de Caso de Uso toma 20 horas-hombre.
- Si el total es 3 o 4, se utiliza el factor de conversión 28 horas-hombre/Punto de Casos de Uso, es decir, un Punto de Caso de Uso toma 28 horas-hombre.
- Si el total es mayor o igual que 5, se recomienda efectuar cambios en el proyecto, ya que se considera que el riesgo de fracaso del mismo es demasiado alto.

El esfuerzo en horas-hombre viene dado por:

$$E = UCP \times CF$$

Donde:

E: esfuerzo estimado en horas-hombre

UCP: Puntos de Casos de Uso ajustados

CF: factor de conversión

Este método proporciona una estimación del esfuerzo en horas-hombre contemplando sólo el desarrollo de la funcionalidad especificada en los casos de uso.

Estimación de la duración del proyecto basado en los Puntos de Casos de Uso

Para una estimación más completa de la duración total del proyecto, hay que agregar a la estimación del esfuerzo obtenida por los Puntos de Casos de Uso, las estimaciones de esfuerzo de las demás actividades relacionadas con el desarrollo de software. Para ello se toma en cuenta el siguiente criterio, que estadísticamente es aceptable, donde se

[1] (Gustav Karner en su tesis en 1993 (Universidad de Linkoping) y supervisado por Ivar Jacobson en Objectory AB. Ha sido analizado posteriormente en otros estudios, como la tesis de Kirsten Ribu (Univerdidad de Oslo) en 2001)

plantea la distribución del esfuerzo entre las diferentes actividades de un proyecto, según la siguiente aproximación:

Tabla 3.24 Estimación duración de un proyecto

Actividad	Porcentaje
Análisis	10.00%
Diseño	20.00%
Programación	40.00%
Pruebas	15.00%
Sobrecarga (otras actividades)	15.00%

Estos valores no son absolutos sino que pueden variar de acuerdo a las características de la organización y del proyecto. Con éste criterio, y tomando como entrada la estimación de tiempo calculado a partir de los Puntos de Casos de Uso, se pueden calcular las demás estimaciones para obtener la duración total del proyecto.

Tabla 3.25 Cuadro Comparativo de las Técnicas de Estimación de Software

CUADRO COMPARATIVO DE LAS METODOLOGÍAS DE ESTIMACIÓN DE SOFTWARE					
Características	**Métrica de Puntos de Función**	**Métrica Bang**	**Medición de Modelos Conceptuales basado en eventos y orientado a objetos**	**Técnica Minería de Datos**	**Técnica Puntos de Casos de Uso**
Objetivo	Medir la funcionalidad del proyecto independientemente de la tecnología de software utilizada.	Medir el alcance y la complejidad del sistema de software. Realizar estimaciones acerca del coste y la duración de procesos.	Realizar el análisis del modelado y de los requisitos del sistema (interfaz de usuario, almacenamiento de datos, flujo de datos, calidad en los requisitos, etc.).	Determinar el tamaño y funcionalidad del software. Obtener la calidad de los productos de software en tiempos de entrega y costos.	Determinar el impacto que los casos de uso producen en la estimación de tamaño, esfuerzo y funcionalidad de proyectos de software.
Fundamento	La obtención de requisitos, siendo útil en cualquiera de las etapas dentro del ciclo	La definición de los requisitos de usuario en las primeras etapas del	La definición de un modelo de dominio como parte de los requisitos de ingeniería en el	La especificación de requisitos del proyecto de software en las	La especificación de Casos de Uso del sistema en base a escenarios,

	de vida del software.	ciclo de vida del software.	desarrollo de un sistema de software.	primeras etapas del ciclo de vida de proyectos.	permitiendo una mejor definición de requisitos del ciclo de vida iterativo-incremental.
Definición	Es una métrica orientada a la función que se centran en la funcionalidad o utilidad del programa.	Es una métrica orientada a la función, que indica el tamaño funcional de un sistema.	Es una técnica útil de modelamiento de dominio orientado a objetos, que se basa en la interacción entre objetos y eventos.	Es una técnica basada en módulos, que sustituye el análisis basado en la verificación de datos por el descubrimiento de información sin necesidad de formular una hipótesis.	Es la estimación del desarrollo de un proyecto, basado en factores funcionales.
Elementos	5 elementos de función: Entradas, Salidas, Consultas Externas, Ficheros Lógicos Internos y Ficheros Externos de Interfaz.	6 componentes principales: Primitivas funcionales, Datos elementales, Objetos, Interrelaciones,	Objetos de dominio, Alfabeto de un objeto y Eventos	Atributos de módulos.	Tipos Actores: simple, medio y complejo. Tipos de Casos de Uso: simple, medio y complejo.

		Estados y Transiciones.			
Formulación de los Factores de Estimación	Factor PFSA-punto de función sin ajustar, PFA-punto de función ajustado, VAF-factor de complejidad y Esfuerzo Requerido.	Factor TC-primitiva funcional, CFP-tamaño funcional orientada a procesos, COB-tamaño funcional orientado a los datos.	Conjunto de Objetos negocio-eventos, Conjunto de Objetos de dominio y tabla OET. Las participaciones de eventos son la base para las estimaciones de gastos y esfuerzo y fijación de precios.	Descripción de los atributos de cada módulo del proyecto.	Factor UUCP-punto de casos de uso sin ajustar, UAW-factor de peso de actores sin ajustar, UUCW-peso de los casos de uso sin ajustar, UCP-puntos de casos de uso ajustados, TCF-factor de complejidad técnica, EF-factor de ambiente, Esfuerzo.
Tipos de estimación de software	Estimación Indicativa o Ball-park y Estimación Consolidada.	Estimación de sistemas orientados a proceso y transformación de datos, estimación de sistemas	Técnica Tabla objeto-evento OET.	Se clasifican en 2 categorías: Supervisados o predictivos y No supervisados o de descubrimiento del	Técnica de Puntos de Función utilizando Casos de Uso.

		orientados a los datos y estimación de sistemas híbrido.		conocimiento.	
Base para la Técnica de Estimación de software	Especificación de Requisitos	Modelo funcional, modelo de datos o modelo de comportamiento.	Modelo Conceptual de datos	Modelo analítico	Datos Históricos
Estimación de la duración del proyecto de software	Conociendo el tamaño funcional del proyecto de software en las primeras etapas del ciclo de vida, puede estimarse la duración total del proceso de desarrollo.	De acuerdo al tamaño del modelo de especificación de requisitos, se determina la duración del proyecto en las primeras etapas del software.	El tamaño, esfuerzo y duración de un proyecto de software se determina de acuerdo al número de elementos que se utilizarán (participación de eventos).	Se determina de acuerdo al tamaño y esfuerzo empleado en las etapas de minería de datos del proyecto de software.	Se requiere: la estimación del esfuerzo obtenida por los Puntos de Casos de Uso y las estimaciones de esfuerzo de las demás fases del ciclo de vida del proyecto.

NOTA: No se considerará a la Técnica de Minería de datos para nuestro análisis, ya que ésta se basa en un modelo analítico para la estimación; y necesitamos basarnos en la Especificación de Requisitos del sistema para la aplicación de cada metodología de estimación de software a nuestro caso práctico.

3.2.9 Caso práctico de aplicación de las técnicas de estimación de software

Para la ejecución de nuestro caso práctico hemos tomado como fundamento la Especificación de Requisitos del Sistema SIGTE (Anexo B). En el Anexo B se desarrolla todas las 4 técnicas de estimación consideradas como caso de estudio, y a continuación se muestra los resultados obtenidos:

Tabla 3.26 Resultados obtenidos caso práctico

COMPARACIÓN ENTRE LAS TÉCNICAS DE ESTIMACIÓN DE SOFTWARE			
Métrica de Puntos de Función	**Métrica Bang**	**Medición de Modelos Conceptuales basado en eventos y orientado a objetos**	**Técnica de Casos de Uso**
PFSA= 422	COB = 52.6	∑ OET= 138	UUCP= 89

Como se puede observar los resultados obtenidos en cada una de las técnicas son empíricos, cada técnica tiene su medida e interpretación, por lo tanto las respuestas son válidas. La forma de comprobar cual de todas las técnicas se acerca más a la realidad, es en base a la comparación con datos históricos existentes.

Para nuestra nueva propuesta metodológica se utilizará el método de conteo de puntos de función de Albrecht, ya que actualmente es el más conocido y empleado.

3.3 El Modelo Cocomo II

COCOMO II, propuesto y desarrollado por Barry Boehm (1989), es uno de los modelos de estimación de costos mejor documentados y utilizados. Boehm dice que *"el modelo permite determinar el esfuerzo y tiempo que se requiere en un proyecto de software a*

partir de una medida del tamaño del mismo expresada en el número de líneas de código que se estimen generar para la creación del producto software"[1].

Debido a la complejidad de los proyectos de software, el modelo original fue modificado, denominándose al modelo actual COCOMO II. El Nuevo modelo permite estimar el esfuerzo y tiempo de un proyecto de software en dos etapas diferentes: diseño temprano y post-arquitectura.

El nuevo modelo permite determinar el esfuerzo y tiempo de un proyecto de software a partir de los puntos de función sin ajustar, lo cual supone una gran ventaja, dado que en la mayoría de los casos es difícil determinar el número de líneas de código de que constará un nuevo desarrollo, en especial cuando se tiene poca o ninguna experiencia previa en proyectos de software. Esto hace que ambos modelos, Puntos de función y COCOMO II, sean perfectamente compatibles y complementarios.

3.3.1 Definiciones

COCOMO II es un modelo que permite estimar el coste, esfuerzo y tiempo cuando se planifica una nueva actividad de desarrollo software. Está asociado a los ciclos de vida modernos. El modelo original COCOMO 81 ha tenido mucho éxito pero no puede emplearse con las prácticas de desarrollo software más reciente tan bien como con las prácticas tradicionales. COCOMO II apunta hacia los proyectos software de los 90 y de la primera década del 2000, y continuará evolucionando durante los próximos años.

3.3.2 Familia de Modelos de Estimación de COCOMO II

Para apoyar a los distintos sectores del mercado software, COCOMO II proporciona una familia de modelos de estimación de coste software cada vez más detallado y tiene en cuenta las necesidades de cada sector y el tipo de información disponible para sostener la

[1] (Boehm, B. and W. Royce (1989), Ada COCOMO and the Ada Process Model, Proceedings, Fifth COCOMO Users' Group Meeting,)

estimación del coste software. Esta familia de modelos está compuesta por tres submodelos, cada uno de los cuales ofrece mayor fidelidad a medida que uno avanza en la planificación del proyecto y en el proceso de diseño.

Estos tres submodelos se denominan:

1. **El modelo de Composición de Aplicaciones.**
2. **El modelo de Diseño anticipado**
3. **El modelo Post-Arquitectura**

3.3.3 Justificación

Se utilizará el modelo de Diseño Anticipado para la aplicación de la metodología ya que se realiza en las primeras etapas del desarrollo en las cuales se evalúan las alternativas de hardware y software de un proyecto. En estas etapas se tiene poca información, lo que concuerda con el uso de Puntos Función, para estimar tamaño y el uso de un número reducido de factores de costo.

3.3.4 El modelo de Diseño anticipado

3.3.4.1 Definiciones

Este modelo puede utilizarse para obtener estimaciones aproximadas del coste de un proyecto antes de que esté determinada por completo su arquitectura. Utiliza un pequeño conjunto de drivers de coste nuevo y nuevas ecuaciones de estimación. Está basado en Punto de Función sin ajustar o KSLOC (Miles de Líneas de Código Fuente).

3.3.4.2 Cálculo del Esfuerzo Nominal

Conversión de Puntos Función a Líneas de Código Fuente (SLOC)

Para determinar el esfuerzo nominal en el modelo COCOMO II los puntos función no ajustados tienen que ser convertidos a líneas de código fuente considerando el lenguaje de implementación (assembler, lenguajes de alto nivel, lenguajes de cuarta generación, etc.). Esto se realiza para los modelos Diseño Temprano y Post Arquitectura teniendo en cuenta la siguiente tabla:

Tabla 3.27 Líneas de código por punto de función de acuerdo al lenguaje (COCOMO II.1999.0)

LENGUAJE	SLOC / UFP [1]
Ada	71
Al Shell	49
APL	32
Assembly	320
Assembly (Macro)	213
ANSI / Quick/Turbo Basic	64
Basic - Compiled	91
Basic Interpreted	128
C	128
C++	29
Programación Orientada a Objetos – (POO)	32

La fórmula para el cálculo es:

Size (SLOC)= UPF x Valor SLOC Lenguaje

Para nuestro modelo hemos seleccionado como lenguaje C#, cuya sintaxis no difiere en mayor grado a la utilizada por java, tomado como base para conversión entre SLOC y UPF = 32 (POO).

[1] (**SLOC / UFP (**Source of Lines Of Code / Unajusted Function Points **))**

El modelo de Diseño Temprano ajusta el esfuerzo nominal usando siete factores de costo.

La fórmula para el cálculo del esfuerzo es la siguiente:

$$PM_{estimado} = PM_{no\min al} \times \prod_{i=1}^{7} EM_i$$

$$PM_{no\min al} = A \times (KSLOC)^B$$

$$B = 1.01 + 0.01 \times \sum_{j=1}^{5} W_j$$

Donde:

***PM*$_{estimado}$** es el esfuerzo Nominal ajustado por 7 factores, que reflejan otros aspectos propios del proyecto que afectan al esfuerzo necesario para la ejecución del mismo.

KSLOC es el tamaño del software a desarrollar expresado en miles de líneas de código fuente.

A es una constante que captura los efectos lineales sobre el esfuerzo de acuerdo a la variación del tamaño, (***A=2.94***).

B es el factor exponencial de escala, toma en cuenta las características relacionadas con la productividad producida cuando un proyecto de software incrementa su tamaño.

EMi corresponde a los factores de costo que tienen un efecto multiplicativo sobre el esfuerzo, llamados Multiplicadores de Esfuerzo (Effort Multipliers).

Factor Exponencial de Escala

El cálculo del Factor Exponencial de Escala ***B*** está basado en factores que influyen exponencialmente en la productividad y esfuerzo de un proyecto de software. Estos factores toman valores dentro de un rango que va desde un nivel ***Muy Bajo*** hasta uno ***Extra Alto***. Cada nivel tiene un peso asociado ***Wj***, y ese valor específico es el que se denomina factor de escala. En la Figura se observan los pesos de cada factor según el nivel, considerados por el software COCOMO II.

Tabla 3.28 Descripción del Factor Exponencial de Escala (COCOMO II.1999.0)

Factor de Escala Wj	Muy Bajo	Bajo	Normal	Alto	Muy Alto	Extra
Precedencia PREC	Completamente sin precedentes	Ampliamente sin precedentes	Algún precedente	Generalmente Familiar	Ampliamente Familiar	Completamente Familiar
Flexibilidad en el desarrollo FLEX	Rigurosa	Relajación Ocasional	Alguna Relajación	Conformidad en general	Alguna Conformidad	Metas generales
Arquitectura/ Resolución de riesgo RESL	Poca (20%)	Alguna (40%)	Siempre (60%)	Generalmente 75%)	Principalmente (90%)	Completo (100%)
Cohesión de equipo TEAM	Interacciones difíciles	Interacciones con alguna dificultad	Interacciones básicamente cooperativas	Ampliamente Cooperativas	Altamente Cooperativas	Interacciones Sin Fisuras
Madurez del proceso PMAT	Desarrollado más adelante					

Figura 3.13 Ponderación del Factor Exponencial de Escala[1]

Multiplicadores de Esfuerzo

A continuación mencionaremos los 7 Multiplicadores de Esfuerzo:

Del Producto

RCPX: Confiabilidad y Complejidad del producto

RUSE: Reusabilidad Requerida de la Plataforma

PDIF: Dificultad de la Plataforma del Personal

PERS: Aptitud del Personal

PREX: Experiencia del Personal del Proyecto

[1] (Software COCOMO II.1999.0)

FCIL: Facilidades

SCED: Cronograma de Desarrollo Requerido

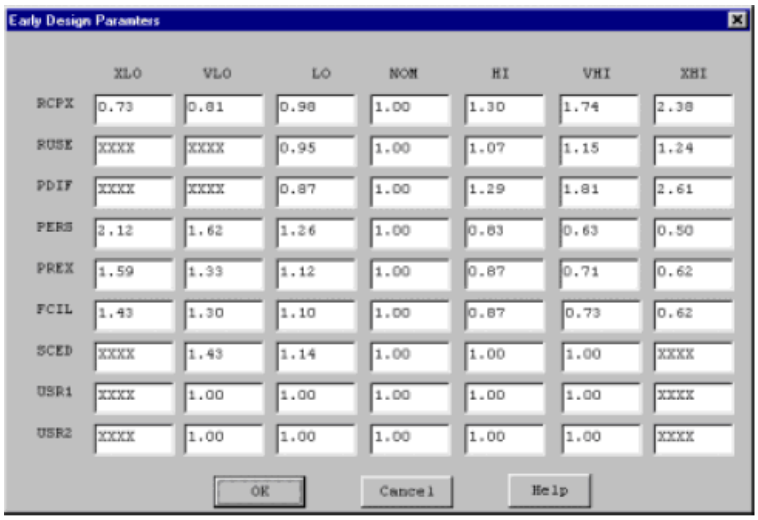


Figura 3.14 Ponderación de los Multiplicadores de Esfuerzo[1]

3.3.4.3 Estimación del Tiempo de Desarrollo

La ecuación inicial para los tres modelos de COCOMO II es:

$$TDEV = \left[3.0 \times PM^{(0.33+0.2\times(B-1.01))}\right] \times \frac{SCED\%}{100}$$

Donde:

TDEV es el tiempo calendario en meses que transcurre desde la determinación de los requerimientos a la culminación de una actividad que certifique que el producto cumple con las especificaciones.

PM* es el esfuerzo expresado en meses personas, calculado sin tener en cuenta el multiplicador de esfuerzo ***SCED.***

B es el Factor de Escala

SCED% es el porcentaje de compresión/expansión del cronograma.

[1] (El Software COCOMO II.1999.0 permite el uso de dos factores del usuario (USR1, USR2) para poder contemplar particularidades de cada proyecto.)

3.4 Desarrollo De Arquitectura En Las Empresas De Software

En la arquitectura centrada en el desarrollo, la planificación es pragmática (Figura 3.15). Los planes de proyectos y el diseño de modelos no usan documentación debido a que su precisión es de corta duración. Una vez que un plan o especificación está fuera de tiempo, es inútil. Por ejemplo, cambios en el código fuente puede rápidamente volver obsoletos los modelos de diseño.

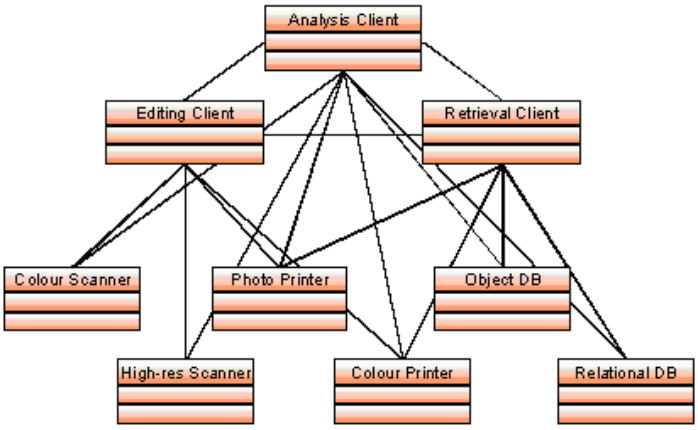


Figura 3.15 Arquitectura de las Empresas de Software

Además, el software los métodos y las normas deben ser tratados como directrices, no mandatos. A los equipos de proyecto se les anima a pensar por sí mismos, y adaptar el proceso para satisfacer las necesidades del proyecto.

La pragmática es un principio fundamental de modelado de software: para las necesidades, la arquitectura y el diseño. Cada modelo tiene un propósito y enfoque. Los modelos deberían documentar las decisiones importantes, sobre la base de supuestos proyectos y prioridades. Decidir lo que es más importante es una habilidad esencial de un arquitecto competente.

3.4.1 Arquitectura centrada en el proceso

La Figura 3.16 muestra los 10 pasos para lograr la Arquitectura Centrada en el Desarrollo, que abarca el ciclo de vida completo del sistema. Este proceso se basa en estándares claves de software.

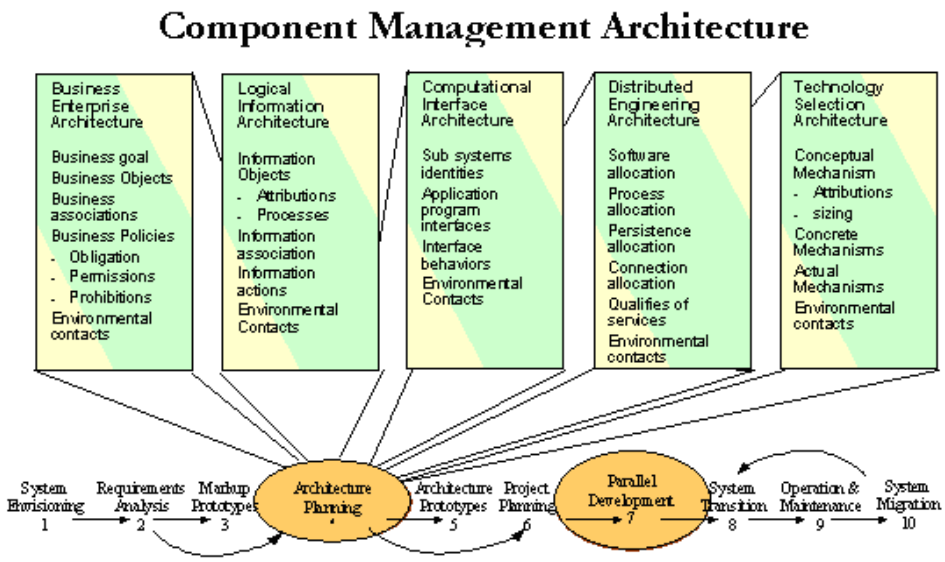


Figura 3.16 Arquitectura Centrada en el Proceso de Desarrollo

Un objetivo fundamental es facilitar la productividad en el paso 7 para el desarrollo iterativo paralelo (es decir, la codificación y las pruebas).

Paso 1: Prevención del Sistema

Al debatir el modelado, se ha mencionado las palabras claves, el enfoque, las hipótesis, y las prioridades. Todos estos son elementos esenciales para la visión del sistema. Si ellos cambian durante el desarrollo del sistema, el proyecto corre el riesgo de ser obsoleto para sus propios modelos. Por lo tanto, el primer paso de la arquitectura centrada en el desarrollo consiste en establecer una visión, que no puede ser cambiada, una vez que comienza el desarrollo (Paso 7). Cualquier cambio debe reflejarse en los planes de proyectos, y particularmente en la arquitectura del sistema (Paso 3).

Paso 2: Análisis de Necesidades

Los requisitos deben definir la conducta externa y la apariencia del sistema, sin el diseño de la estructura interna del sistema. El comportamiento externo incluye acciones internas (como la persistencia o cálculo) que se requieren para garantizar el comportamiento exterior deseado. La apariencia externa comprende el diseño y la navegación por las pantallas de interfaz del usuario.

Un planteamiento eficaz para la captura de requisitos de comportamiento es a través de casos de uso. Un caso de uso consta de un alto nivel gráfico y una amplia descripción textual. Los casos de uso son eficaces para expresar conceptos complejos. Por lo tanto, es ideal para garantizar la simplicidad y la claridad en la representación de nivel superior correspondientes a los requisitos conceptuales.

La apariencia, funcionalidad y navegación de la interfaz de usuario está estrechamente relacionado con los casos de uso. Un planteamiento eficaz para la definición de las pantallas se llama "baja fidelidad de prototipos". En este enfoque, las pantallas se han extraído a cabo con papel y lápiz. Una vez más, los usuarios finales expertos de dominio se encuentran involucrados en el proceso de definición de pantalla.

Con los casos de uso y las interfaces de usuario definido, se ha creado el contexto para la planificación arquitectónica. Además de generar la documentación, el equipo de arquitectura adquiere un profundo conocimiento de las capacidades del sistema deseada en el contexto del usuario final de dominio.

Un producto final de análisis de requisitos es un proyecto que debería ampliarse durante la planificación de la arquitectura (Paso 3).

Paso 3: Planificación de la Arquitectura

Arquitectura es el puente de la enorme brecha entre las necesidades y los programas informáticos. Los requisitos son inherentemente ambiguos, intuitivos, e informales. Es la parte izquierda del cerebro. El software, por otra parte, tiene características opuestas. El código fuente del software es una notación formal. El software es interpretado sin ambigüedades por una máquina, y su significado es, lógicamente, no intuitivo (es decir, difícil de descifrar). Es el lado derecho del cerebro.

La primera función de la arquitectura es definir el mapeo entre estos dos extremos. La arquitectura capta la parte intuitiva de las decisiones en una manera más formal (lo cual es útil para los programadores), y define la estructura interna del sistema antes de que sea difícil en el código (de modo que los requisitos actuales y futuros pueden ser satisfechos).

La arquitectura es un plan que gestiona la complejidad del sistema de una forma que permite la construcción del sistema y da cabida a cambios. La arquitectura tiene otra función importante definir la organización del proyecto de software. (Vea el Paso 6).

La Planificación de la Arquitectura es el paso clave que falta dentro de muchos proyectos de software actuales, procesos y métodos.

ODP[1] es una poderosa manera de pensar acerca de sistemas complejos, lo que simplifica la toma de decisiones (es decir, más inteligente, no más difícil). Organiza la arquitectura del sistema en términos de cinco puntos de vista estándares, que describe aspectos importantes del mismo sistema. Estos incluyen los puntos de vista empresarial, la información lógica, interfaz computacional, ingeniería distribuida, y la selección de tecnologías (Figura 3.17).

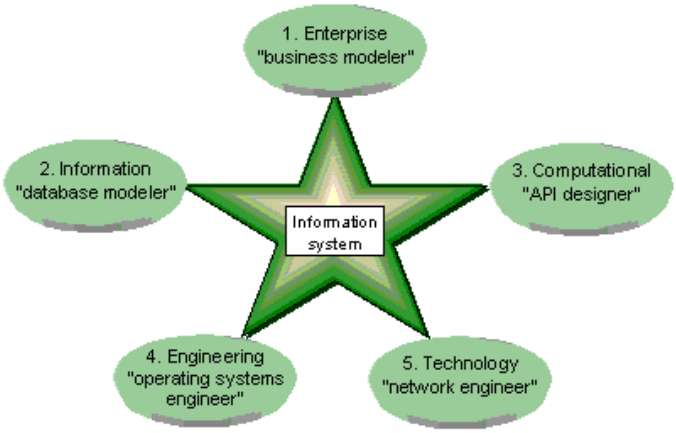


Figura 3.17 Puntos de Vista ODP

Para cada punto de vista es importante determinar la conformidad con los requisitos de arquitectura. Si la conformidad no tiene una definición objetiva, entonces la arquitectura no tiene sentido, porque no tienen claro el impacto en la aplicación. La ODP facilita este proceso debido a una generalizada conformidad.

[1] (ODP, Open Distributed Processing)

Paso 4: Mockup

Las pantallas definidas en la etapa 2 se utilizan para crear una línea mockup del sistema. Los datos y simples archivos IO[1] pueden utilizarse para proporcionar una simulación más realista de interfaz de usuario. El mockup se demuestra a los usuarios finales y la gestión de patrocinadores.

Los usuarios finales y los arquitectos deberían examinar conjuntamente la ejecución de los casos de uso (paso 2) con el fin de validar los requisitos. A través de la mockup, la gestión es capaz de ver progreso visible, un logro político importante para la mayoría de los proyectos. Este paso es un ejemplo de un control externo (o vertical) de incremento, que se utiliza para la reducción de los riesgos.

Paso 5: Arquitectura de prototipos

La arquitectura centrada en el desarrollo de software se ocupa de los grandes riesgos por adelantado, tales como los riesgos de integración y desempeño del sistema de riesgos. Los arquitectos se refieren a toda las conductas del sistema y problemas que pueden "hacer o deshacer" el sistema. Como resultado de ello, un objetivo de la arquitectura es para evitar grandes y costosos errores en el desarrollo de sistemas y que, muchas otras disciplinas de TI[2] se centran en "no hacer pequeños errores". El propósito de la arquitectura prototipos es resolver grandes riesgos en una fase temprana del desarrollo del ciclo de vida del software.

El prototipo de la arquitectura se utiliza para validar la ingeniería computacional y arquitecturas, incluyendo el flujo de control y el calendario de distribución.

El uso de tecnologías como CORBA, especifica una arquitectura computacional que puede ser automáticamente compilada en un conjunto de archivos (headers) y esqueletos (skeletons). El código se inserta en los esqueletos para simular la transformación.

[1] (IO, InputOutput / EntradaSalida)
[2] (TI, Tecnologías de Información)

La arquitectura de prototipos puede ser utilizado para crear un modelo arquitectónico para la construcción del sistema.

Paso 6: Planificación de la Gestión de Proyectos

Los planes de gestión de proyectos son definidos y validados para resolver cuestiones relacionadas con los recursos, incluyendo la asignación de personal, instalaciones, equipo, tecnología comercial y las adquisiciones. El calendario y el presupuesto están establecidos en función de la disponibilidad (tiempo) para los recursos y las actividades del proyecto.

El calendario para el paso 7 está previsto en términos de actividades paralelas para externos e internos incrementos. Los incrementos externos (centrado en la funcionalidad visible del usuario) apoyan la reducción de los riesgos con respecto a los requisitos y el apoyo a la gestión (ver paso 4). Los incrementos internos apoyan el uso eficiente de los recursos para el desarrollo, por ejemplo, el desarrollo de back-end[1], servicios utilizados por múltiples subsistemas.

La arquitectura centrada en el proceso permite incrementos paralelos. Debido a que el sistema está definido por los límites de cómputo, los equipos de desarrollo trabajan de manera independiente, pero pueden trabajar en paralelo con otros equipos, dentro de sus límites asignados. La integración de la planificación incluye incrementos, que abarcan los límites de arquitectura.

La actividad final de la planificación de la gestión de proyectos de arquitectura es el examen y la decisión. Las empresas patrocinadoras han hecho relativamente pocos compromisos, en comparación con la escala completa de desarrollo (alrededor del 5% del coste del sistema, según el proyecto).

[1] (**Front-end** y **back-end** son términos que se relacionan con el principio y el final de un proceso.)

Paso 7: Desarrollo Paralelo Incremental

Cada incremento implica un proceso de desarrollo, incluyendo el diseño, codificación, y pruebas. Inicialmente, la mayoría de los incrementos se centrará en los distintos subsistemas. A medida que el proyecto avanza, habrá un mayor número de los incrementos que involucran las múltiples integraciones del subsistema.

Para la mayor parte de la actividad de desarrollo de software, la arquitectura es congelada, excepto en algunos puntos previstas, y en las actualizaciones de arquitectura donde se pueden insertar sin interrupción. La estabilidad de arquitectura permite el desarrollo paralelo.

Paso 8: Sistema de Transición

Un importante papel del arquitecto en este paso implica la aceptación del sistema. El arquitecto debería confirmar que el sistema de aplicación es conforme con las especificaciones y satisface las necesidades de los usuarios. Esta tarea se llama arquitectura de certificación.

En efecto, el arquitecto debe ser un árbitro imparcial entre los intereses de los usuarios finales y los de los desarrolladores del sistema. Si los usuarios finales definen nuevos requisitos, el arquitecto evalúa la solicitud, y trabaja con ambas partes para planificar soluciones viables.

Paso 9: Operaciones y Mantenimiento

Este paso es importante dentro de la arquitectura centrada en el desarrollo. Sea o no el resultado de "un software bien hecho" que sea eficaz, en este paso será demostrado. La mayoría de los gastos del sistema se realizarán en esta fase. Y tanto como el 70% del coste se debe a las extensiones del sistema y los cambios tecnológicos que son la fuente clave de desarrollo continuo.

Paso 10: Sistema de las Migraciones

El sistema de la migración a un seguimiento de la arquitectura seleccionada se produce cerca del final del ciclo de vida del sistema. Dos procesos importantes para la migración del sistema se llaman big bang y Chicken Little. Un big bang es una sustitución rápida para evitar errores. Mientras el Chicken Little tiene un enfoque más eficaz, y, en definitiva, más éxito. Chicken Little supone una operación entre los objetivos y los sistemas heredados. El objetivo inicial usuarios de la red son el grupo piloto (como en el paso 8). Y los pasos 8, 9 y 10 forman parte de un proceso continuo de la migración.

3.5 Herramienta CodeSmith Studio

CodeSmith es un generador de código a base de plantilla que permite generar el código para cualquier lengua de texto. El código generado puede ser personalizado por el empleo de propiedades. Una propiedad puede ser cualquier objeto de .NET que tiene un diseñador y puede ser tan simple como una propiedad booleana que le permite condicionalmente añadir o quitar el código del resultado, a un objeto como el TableSchema (incluido en SchemaExplorer) que proporciona el acceso a base de datos. CodeSmith viene con muchos tipos de propiedades estándar y es el 100 % extensible por que le permite al usuario crear tipos de propiedades distintas.
La sintaxis de CodeSmith es casi idéntica a ASP.NET. Es tan familiar que rápidamente se puede aprender la sintaxis de plantilla. Se puede usar C#, VB.NET o JSCRIPT.

3.5.1 Características Generales

- CodeSmith Estudio hace el edificio de sus propias plantillas de forma muy fácil.
- Puede compilar y ejecutar sus plantillas muy rápido y de manera eficiente.
- Es fácil para eliminar fallos de las plantillas, ya que tiene un revelador de plantilla.

- Permite fácilmente usar XML[1] para conducir sus plantillas. Si proporcionan un esquema XSD[2], un modelo de objeto de XML fuerte escrito a máquina, automáticamente será generado y hace el funcionamiento con XML simple. Si no proporcionan un esquema XSD CodeSmith le permitirá trabajar directamente con un caso XmlDocument.
- Las capacidades que se combinan en CodeSmith le permiten combinar el código generado y escrito a mano dentro de un archivo solo.
- El cliente de consola puede fácilmente automatizar la generación de código y puede ejecutar plantillas una por una o juntas.
- Las Plantillas que generan escrituras SQL pueden ser auto ejecutadas después de la generación que tiene el despliegue fácil en cuenta de código de SQL generado.
- CodeSmith incluye muchos rasgos más todo diseñado para ayudar a escribir el código más rápido y con menos defectos.

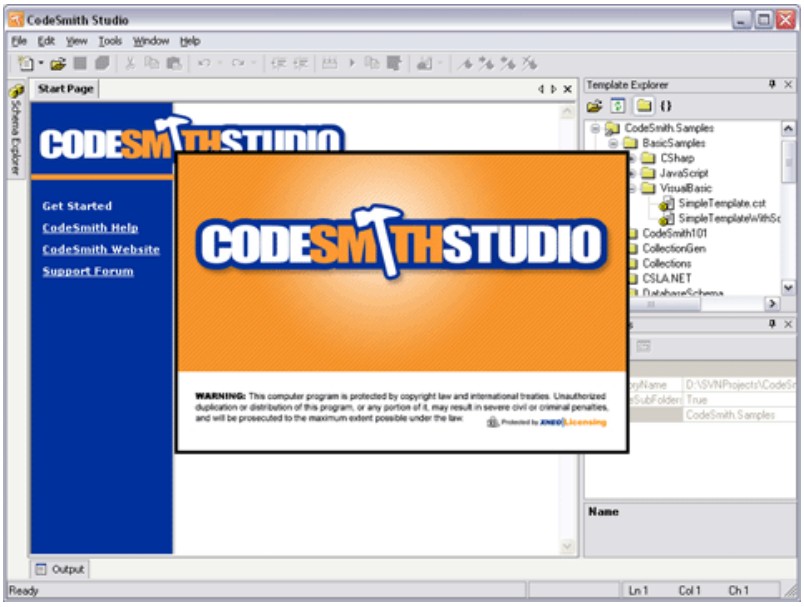


Figura 3.18 Interface de CodeSmith Studio

[1] (XML, sigla en inglés de Extensible Markup Language (lenguaje de marcas ampliable). XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable - http://es.wikipedia.org/wiki/XML)

[2] (Estándar para la expresión de comprobantes fiscales digitales - tp://ftp2.sat.gob.mx/asistencia_servicio_ftp/publicaciones/solcedi/cfdv2.xsd)

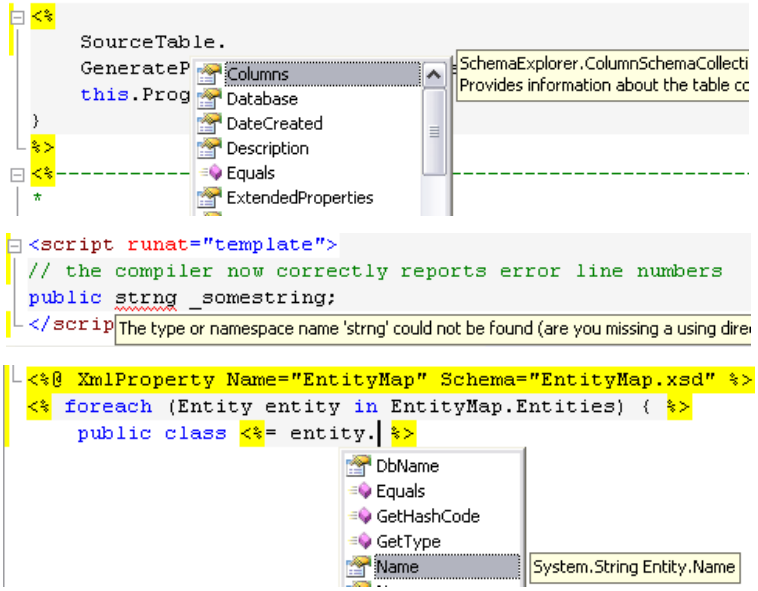


Figura 3.19 Uso CodeSmith Studio

4 CAPÍTULO IV - MPE (Model Prototype Estimation)

4.1 MPE: Metodología Basada En Modelos Conceptuales, Puntos De Función Y Líneas De Código

El método empleado para determinar el esfuerzo necesario introducido por la arquitectura en el desarrollo de proyectos de software, se basa en la aplicación de la siguiente secuencia de pasos:

1. Definición del modelo conceptual.
2. Cálculo de PF basados en el modelo conceptual.
3. Determinación del número de LOC producidas por punto de función en base a la técnica de PF.
4. Definición de la arquitectura de software.
5. Generación de un prototipo basado en la definición de la arquitectura para un framework[1] específico.
6. Conteo del número de LOC del prototipo.
7. Cálculo del esfuerzo por arquitectura y por funcionalidad mediante el modelo COCOMO.

[1] (Es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado - http://es.wikipedia.org/wiki/Framework)

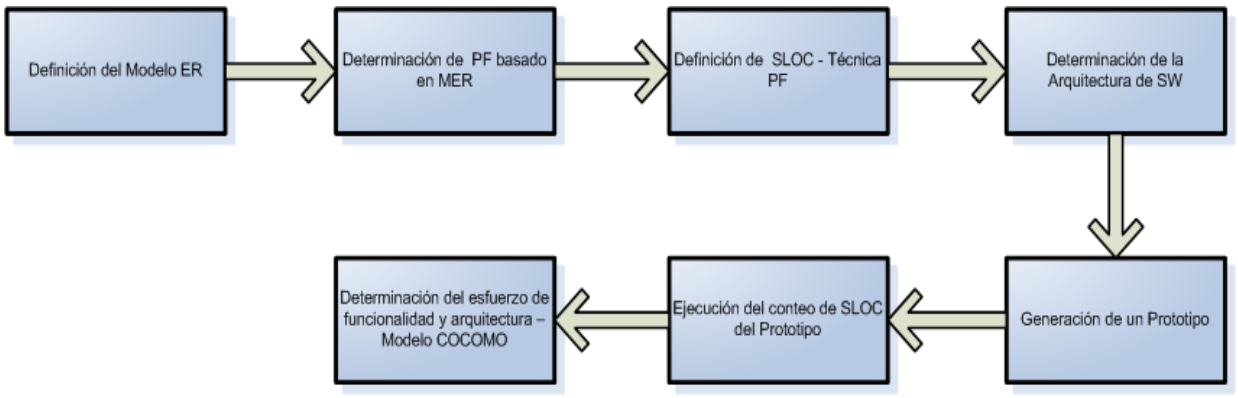


Figura 4.1 Estructura de Procesos de MPE

A continuación se describen cada uno de ellos.

4.1.1 Definición del Modelo Conceptual

Los modelos conceptuales son una técnica muy útil durante el proceso de especificación de requisitos, básicamente permiten modelar el dominio del problema mediante el uso de una ontología[1].

En la literatura enfocada al análisis de requisitos, se distinguen entre modelos conceptuales y modelos de sistema, diferenciándolos en que los primeros describen el universo del discurso, en tanto que los segundos modelan el futuro sistema a construirse; por lo que, en el estudio presentado es importante mencionar que se utilizaron modelos de sistema reales, ya que el desarrollo de prototipos exigen un alto grado de precisión y por tanto formalismo para ser realmente útiles.

El uso del MER (Modelo Entidad Relación) como modelo conceptual ha sido seleccionado para el desarrollo del método; éste permite abstraer las entidades y sus asociaciones en el dominio del discurso, generando una vista del sistema que puede ser interpretada desde la perspectiva de los puntos de función, si se incorpora a cada entidad, las operaciones básicas que pueden ser realizadas sobre ellas. Dichas operaciones

[1] (La formulación de un exhaustivo y riguroso esquema conceptual dentro de un dominio dado, con la finalidad de facilitar la comunicación y la compartición de la información entre diferentes sistemas - http://es.wikipedia.org/wiki/Ontolog%C3%ADa_(Inform%C3%A1tica))

conocidas como CRUD (Create, Read, Update, Delete), serán la base para la cuenta de los Puntos de Función y la generación del prototipo que las implemente.

4.1.2 Cálculo de puntos de función (PF) basados en el modelo conceptual

La determinación de PF se basa en el método de conteo de Albrecht, con la variante de que se ha tomando como fuente de análisis al MER en lugar de la especificación de los requisitos funcionales del sistema. Aunque si se aplica formalmente la técnica propuesta por Albrecht, esto constituye una aproximación a la medición del sistema dependiente de la tecnología antes que a la medición funcional desde la perspectiva del usuario, pero se justifica en razón de que finalmente el prototipo generará automáticamente el número de líneas de código teniendo al MER como su origen. Este proceso se resume a lo siguiente:

Cada entidad del modelo se cuenta como un ILF[1]. Para determinar la complejidad, se cuenta cada atributo como un DET[2] que posee un único RET[3]. Se aplica la tabla propuesta en el método de conteo de puntos de función para los ILFs.

Tabla 4.1 Definición de complejidad para ILFs

Tipos de Registros (RET)	Elementos de Datos (DET)		
	1-19	20-50	>50
1	Baja	Baja	Media
2-5	Baja	Media	Alta
>5	Media	Alta	Alta

Por cada entidad, se cuentan las operaciones de Creación, Actualización y Eliminación como tres EI[4].

[1] (Archivo Lógico Interno)
[2] (Tipo de Dato Elemental)
[3] (Grupo de Datos Relacionados)
[4] (Entradas Externas)

Se determina la complejidad para las EI contando cada atributo de la entidad como un DET y sumando un DET adicional por el elemento que desencadena la acción[1]. Se cuentan el número de relaciones por cada entidad y se obtienen el número de FTR[2].

Tabla 4.2 Definición de complejidad para EI

Archivos referenciados FTR	Elementos de Datos DET		
	1-4	5-15	>15
0-1	Baja	Baja	Media
2	Baja	Media	Alta
3 ó más	Media	Alta	Alta

Por cada entidad, se cuenta una EQ[3] como operación de lectura.

Se determina la complejidad para las EQ de la misma forma como se define en las EI, pero a diferencia de éstas, se aplica la siguiente tabla:

Tabla 4.3 Definición de complejidad para EQ

Archivos referenciados FTR	Elementos de Datos DET		
	1-5	6-19	>19
0-1	Baja	Baja	Media
2-3	Baja	Media	Alta
>3	Media	Alta	Alta

[1] (Botón de comando en la interfaz de usuario)
[2] (Tipos de Archivos Referenciados)
[3] (Consulta Externa)

Se asume que el sistema no tendrá EIF[1] y tampoco se generarán EO[2].

Finalmente, se determina el total de UFP[3] sumando y ponderando los resultados en base a la tabla definida por el método de puntos de función.

Hay que señalar que en este punto se han presentado los aspectos más relevantes para la aplicación del método de Albrecht con la variante que aquí se plantea, en virtud de que existe suficiente literatura al respecto.

4.1.3 Determinación del número de líneas de código producidas por punto de función en base a la técnica de PF

La técnica de Puntos de Función, establece un mecanismo denominado "backfiring"[4]. Esto conduce a la utilización de una tabla que es el punto donde convergen la funcionalidad y la tecnología definida por un lenguaje de desarrollo. De esta decisión tecnológica, depende en mayor grado el esfuerzo requerido para implementar la funcionalidad de la aplicación; así en la siguiente tabla podemos apreciar los equivalentes entre puntos de fundón sin ajustar y cantidad de líneas de código estimados para diferentes lenguajes de programación (ver Tabla 3.27).

La fórmula para el cálculo es:

Size (SLOC)= UPF x Valor SLOC Lenguaje

Para nuestro modelo hemos seleccionado como lenguaje C# de Microsoft, cuya sintaxis no difiere en mayor grado a la utilizada por java de Sun, tomado como base para conversión entre SLOC y UPF = 32 (POO).

[1] (Archivos de Interfaces Externas)
[2] (Salidas Externas)
[3] (Puntos de Función sin ajustar)
[4] (Transformación entre líneas de código por cada PF)

Cálculo de SLOC para la Arquitectura del Sistema

Para el cálculo de la arquitectura del sistema se usa una herramienta que generará automáticamente el código de todo el sistema de acuerdo al lenguaje de programación a utilizar (en nuestro caso de estudio Lenguaje C#); para determinar el tamaño de la arquitectura del sistema, se obtendrá realizando el siguiente cálculo:

$$ALOC = TLOC - BLOC$$

Donde:

BLOC = Líneas de código del negocio (Bussines Lines of Code)
ALOC = Líneas de código de arquitectura (Architecture Lines of Code)
TLOC = Total de líneas de código del sistema (Total Lines of Code)

4.1.4 Definición de la arquitectura del software

La arquitectura de software consiste en un conjunto de patrones, prácticas y abstracciones coherentes que proporcionan el marco de trabajo necesario para guiar el desarrollo de soluciones de software. Para el presente estudio se ha definido un marco de trabajo (Framework) genérico en el cual se engloban cuatro dominios de arquitectura:

1. Arquitectura de Negocio, definiendo el "governance"[1], la organización y los procesos de negocio implicados.
2. Arquitectura de Aplicaciones, proveyendo una guía de implementación de cada una de las aplicaciones consideradas, así como de las interacciones entre los sistemas y las áreas / procesos de negocio.
3. Arquitectura de Datos, contemplando las estructuras lógicas y físicas de datos, además de los recursos de gestión.
4. Arquitectura de Tecnología, que describe la infraestructura software que debe soportar las aplicaciones.

[1] (La Estrategia del Negocio)

4.1.5 Generación de un prototipo basado en la definición de la arquitectura para un framework específico

Con la finalidad de automatizar los aspectos clave de todo el proceso de desarrollo de un producto de software, como son por ejemplo mecanismos de acceso a datos, seguridad, comunicaciones entre otras, los prototipos se implementan a través de una herramienta CASE[1], la misma que genera un conjunto de clases que aseguran el cumplimiento de las guías de arquitectura más comúnmente utilizadas en soluciones empresariales de alta disponibilidad requeridas actualmente por la industria.

Para la generación del prototipo se toma como entrada un modelo Entidad / Relación y en base a este se aplica el ORM[2], que es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional. En la práctica esto crea una base de datos orientada a objetos virtual, por sobre la base de datos relacional. Esto posibilita el uso de las características propias de la orientación a objetos, además se provee mecanismos de acceso a datos robustos e interfaces de servicios para exponer la funcionalidad de la aplicación independientemente del protocolo utilizado.

Generación del Prototipo

Para la generación del Prototipo se utiliza la herramienta CodSmith Studio. Se selecciona el Modelo Conceptual y las tablas correspondientes para la generación de código por defecto de la arquitectura base del sistema, para nuestro caso de estudio el Framework utiliza el lenguaje C#.

Se compila el código y se genera un archivo *.sln,* que tendrá todo el código de la arquitectura del sistema.

[1] (Computer Aided Software Engineering)
[2] (Mapeo Objeto Relación)

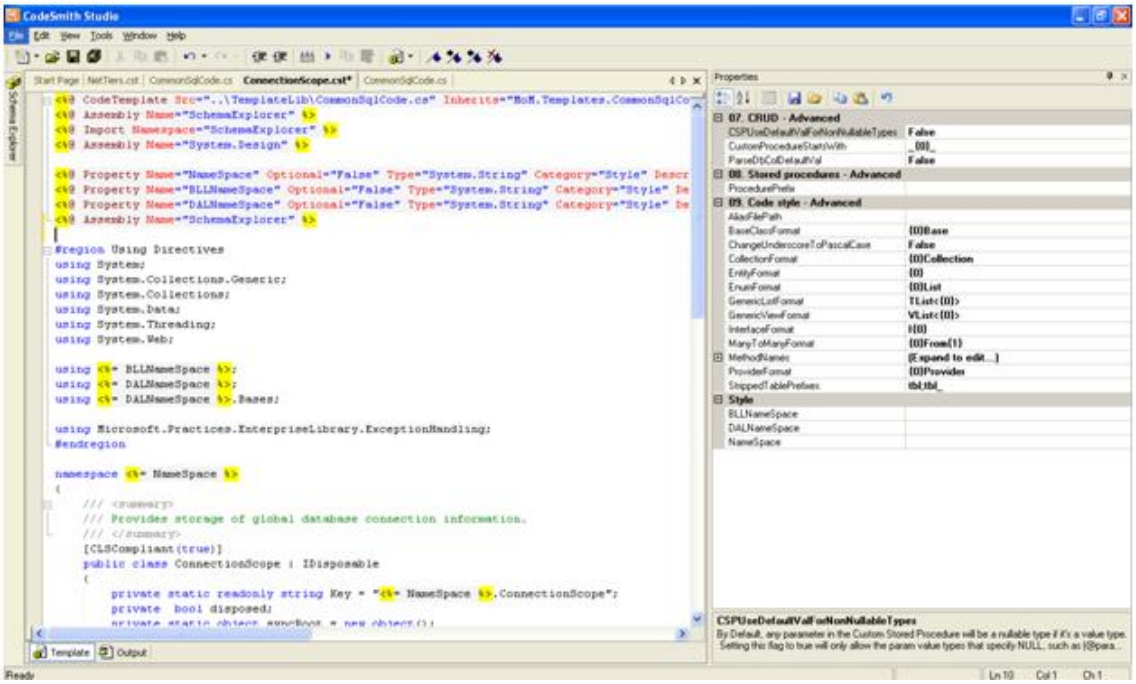


Figura 4.2 Generación del Prototipo

Las siguientes Figuras muestran la aplicación para dar soporte a la automatización del método aquí expuesto.

Dentro del software modelo MPE, se presenta la pantalla inicial y la pantalla de ingreso del modelo Entidad Relación, donde seleccionamos el archivo *.sln* que tiene la generación de la arquitectura del sistema, para convertirla en líneas de código.

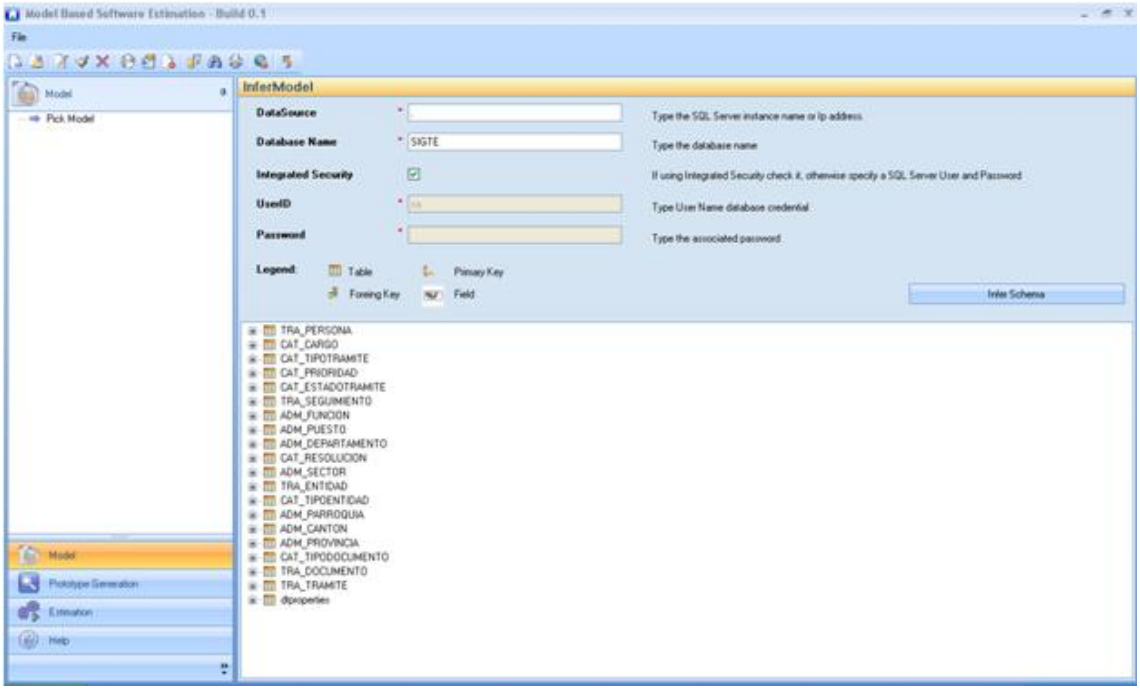


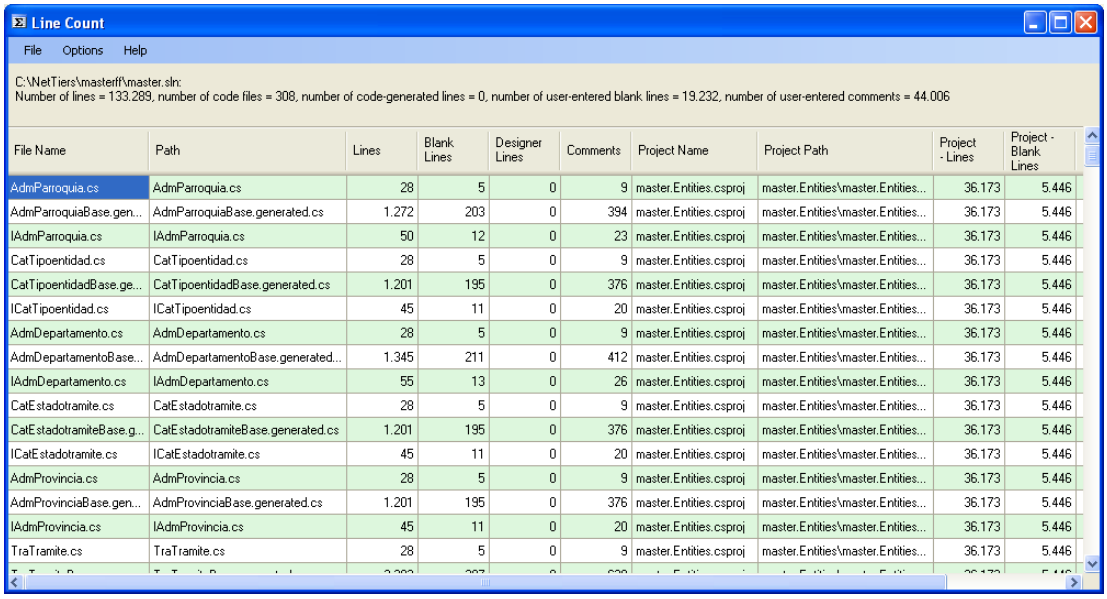
Figura 4.3 Selección y Descripción de tablas del Modelo Entidad Relación

4.1.6 Conteo del número de líneas de código del Prototipo

De acuerdo con Fenton y Pfleeger, la definición más extendida de "Línea de Código" es la propuesta por Hewlett-Packard[1]: cualquier sentencia, salvo las de comentario y las líneas en blanco. Para el método expuesto, se ha tomado esta definición para la determinación del total de líneas de código generadas en el prototipo. Las Figuras siguientes, muestran las pantallas del conteo de de número de líneas de código obtenidos por la herramienta que aplica la metodología.

A continuación se realiza la descripción del total de líneas de código que tiene el sistema, total de líneas comentadas, en blanco y archivos de código.

[1](Conocida como HP, es la mayor empresa de tecnologías de la información del mundo. Fabrica y comercializa hardware y software además de brindar servicios de asistencia relacionados con la informática - http://es.wikipedia.org/wiki/Hewlett_Packard)



File Name	Path	Lines	Blank Lines	Designer Lines	Comments	Project Name	Project Path	Project - Lines	Project - Blank Lines
AdmParroquia.cs	AdmParroquia.cs	28	5	0	9	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
AdmParroquiaBase.gen...	AdmParroquiaBase.generated.cs	1.272	203	0	394	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
IAdmParroquia.cs	IAdmParroquia.cs	50	12	0	23	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
CatTipoentidad.cs	CatTipoentidad.cs	28	5	0	9	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
CatTipoentidadBase.ge...	CatTipoentidadBase.generated.cs	1.201	195	0	376	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
ICatTipoentidad.cs	ICatTipoentidad.cs	45	11	0	20	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
AdmDepartamento.cs	AdmDepartamento.cs	28	5	0	9	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
AdmDepartamentoBase...	AdmDepartamentoBase.generated...	1.345	211	0	412	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
IAdmDepartamento.cs	IAdmDepartamento.cs	55	13	0	26	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
CatEstadotramite.cs	CatEstadotramite.cs	28	5	0	9	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
CatEstadotramiteBase.g...	CatEstadotramiteBase.generated.cs	1.201	195	0	376	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
ICatEstadotramite.cs	ICatEstadotramite.cs	45	11	0	20	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
AdmProvincia.cs	AdmProvincia.cs	28	5	0	9	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
AdmProvinciaBase.gen...	AdmProvinciaBase.generated.cs	1.201	195	0	376	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
IAdmProvincia.cs	IAdmProvincia.cs	45	11	0	20	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
TraTramite.cs	TraTramite.cs	28	5	0	9	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446

Figura 4.4 Conteo de líneas de código.

El software de soporte desarrollado permite una validación empírica más eficaz del método; se ha conseguido determinar los indicadores necesarios en el menor tiempo y precisión posible.

4.1.7 Cálculo del esfuerzo por arquitectura y por funcionalidad mediante el modelo COCOMO

Una de las actividades cruciales en la gestión de un proyecto de software, es la estimación del esfuerzo y tiempo de desarrollo. El método planteado usa el modelo COCOMO[1], para establecer estos indicadores.

La ecuación básica del modelo COCOMO es la siguiente:

$$MP_{Nominal} = A \times (Size)^{B}$$

[1] (COnstructive COst MOdel, desarrollado por Boehm en 1981)

Donde:

MP = Esfuerzo Nominal en meses-persona.

A = Constante

Size = Tamaño del software medido en líneas de código fuente.

B = Factor de escala

Para el conteo de Puntos de Función, no se desarrolla el proceso de estimación de esfuerzo, tiempo y personal mediante el modelo COCOMO, en virtud de que la técnica es por demás conocida; sin embargo es preciso señalar que para efectos de no introducir nuevas variables en el proceso, se ha tomado un valor nominal tanto para los factores de escala (B) como para los drivers de costo (se consideran igual a 1, Esfuerzo Nominal). Finalmente el modelo usado es COCOMO II en su versión de Diseño Anticipado, aunque bien podría utilizarse el modelo Post-Arquitectura.

Para definir el esfuerzo total del sistema se conseguirá con el siguiente cálculo:

$$ETS = EFS + EAS$$

ETS= (Esfuerzo total del sistema) Effort Total System

EFS= (Esfuerzo functional del sistema) Effort Functional System

EAS= (Esfuerzo de la arquitectura del sistema) Effort of Architecture System

4.2 Interpretación Y Análisis De Resultados

El método ha sido aplicado a tres proyectos reales de software:

- SIGTE: Sistema de Gestión de Trámites. (ANEXO C)
- SAPIV: Sistema de Administración de Proyectos de Investigación. (ANEXO D)
- VENTAS: Sistema de gestión de Ventas. (ANEXO E)

En todos los casos, se utilizan los términos “Bussines” para hacer referencia a la lógica de negocio o funcionalidad del sistema y “Arquitecture” para denotar la arquitectura. Los resultados obtenidos se detallan a continuación:

Tabla 4.4 Líneas de código generadas

	SIGTE		SAPIV		VENTAS	
	LOC	%	LOC	%	LOC	%
BLOC	13344	19	18048	14	32320	24
ALOC	57359	81	87791	86	103304	76
TLOC	70703	100	105839	100	135624	100

BLOC = Bussines Lines of Code

ALOC = Architecture Lines of Code

TLOC = Total Lines of Code

TLOC = BLOC + ALOC

Del análisis de la Tabla 4.4 se desprende que en todos los casos el porcentaje de líneas de código generados por arquitectura (ALOC), supera al porcentaje obtenido por la implementación de funcionalidad (BLOC). Siendo SIGTE, el sistema de menor tamaño en líneas de código (TLOC) seguido por SAPIV y VENTAS.

Tabla 4.5 Puntos de función de las aplicaciones analizadas

	SIGTE		SAPIV		VENTAS	
	LOC	%	LOC	%	LOC	%
BFP	417	19	564	14	1010	24
AFP	1793	81	2743	86	3228	76
TFP	2210	100	3307	100	4238	100

BFP = Puntos de función sin ajustar del negocio (Bussines Function Points)
AFP = Puntos de función sin ajustar de la arquitectura (Architecture Function Points)
TFP = Total de puntos de función (Total Function Points)

La tabla 4.5 indica la cantidad de puntos de función sin ajustar tanto en funcionalidad como en arquitectura, como era de suponer, al existir en la técnica de Albrecht una relación lineal entre líneas de código y puntos de función, se desprende la misma proporción de funcionalidad respecto a arquitectura. SIGTE tiene la menor cantidad de puntos de función sin ajustar y Ventas la mayor cantidad.

Tabla 4.6 Tamaño de la base de datos de las aplicaciones analizadas

	SIGTE	SAPIV	VENTAS
DBS	19	25	45
REL%	4.30	6.34	3.20

DBS = Tamaño de la base de datos (Data Base Size)
REL % = Porcentaje Relativo entre arquitectura y funcionalidad.

La Tabla 4.6 muestra la relación entre el tamaño del modelo Entidad Relación medido como el número de tablas de la base de datos (DBS), respecto al porcentaje relativo (REL%) entre arquitectura y funcionalidad. Es importante destacar que para todos los casos, la arquitectura aproximadamente cuadriplica a la funcionalidad.

Tabla 4.7 Estimación esfuerzo, tiempo y personal de las aplicaciones analizadas

	ESFUERZO			TIEMPO			PERSONAL		
	BLE	ARE	TOT	BLT	ART	TOT	BLS	ARS	TOT
SIGTE	65.87	379.02	444.89	13.78	23.96	37.74	4.7	15.8	20.5
SAPIV	94.63	631.6	726.23	15.5	28.15	43.65	6.1	22.4	28.5
VENTAS	190.42	767.87	958.29	19.3	29.95	49.25	9.8	25.6	35.4

BLE = Esfuerzo en la lógica de negocio (Bussines Logic Effort)

ARE = Esfuerzo de arquitectura (Architecture Effort)

TOT = Total de esfuerzo del sistema

BLT = Tiempo en la lógica de negocio (Bussines Logic Time)

ART = Tiempo en arquitectura (Architecture Time)

BLS = Personal en la lógica de negocio (Bussines Logic Staff)

ARS = Personal en la arquitectura (Architecture Staff)

La determinación del esfuerzo, tiempo y personal mediante el modelo COCOMOII mostrados en la Tabla 4.7, confirman la estrecha relación entre la cantidad de líneas de código y puntos de función de los proyectos.

Se puede observar que hay ligeras variaciones en cuanto al tiempo que requiere cada proyecto, esto es razonable debido a la diferencia total en el recurso humano a utilizar.

5 CAPITULO V - APLICACIÓN EN CASOS PRÁCTICOS

5.1 Definición De Los Casos Prácticos

Se han utilizado casos reales para la validación empírica del método. Los tres casos prácticos que fueron aplicados con nuestro nuevo prototipo de estimación son:

- SIGTE: Sistema de Gestión de Trámites. (ver ANEXO D)
- SAPIV: Sistema de Administración de Proyectos de Investigación. (ver ANEXO E)
- VENTAS: Sistema de gestión de Ventas. (ver ANEXO F)

Se realizó el caso práctico del sistema SIGTE mediante la Especificación de Requisitos, definiendo posteriormente el modelo conceptual; mientras que el sistema SAPIV y de VENTAS, se definieron mediante el modelo conceptual de datos respectivo.

5.2 Lecciones Aprendidas

- Definir bien el modelo conceptual de datos para la aplicación de la nueva metodología, para obtener resultados más reales de estimación.

- Si consideramos el documento de especificación de requisitos como base para la creación del modelo conceptual, puede dar motivo a diversas interpretaciones entre los planificadores encargados del desarrollo de la estimación.

- La aplicación manual de la metodología es muy tediosa, por esta razón es preciso disponer de un entorno automático para aplicarla. Por ello se ha implementado la herramienta MPE, que se presenta en el capítulo siguiente.

- Gracias a la utilización de la herramienta automática MPE, se evita mayor porcentaje de error en los resultados obtenidos, ya que la aplicación manual puede provocar confusión o equivocaciones en el uso de fórmulas.

- Dentro de las técnicas de estimación de software existentes, el único aspecto considerado es la funcionalidad del sistema; lo que nunca ha permitido tener datos cercanos a la realidad; y por esta razón ninguna de estas técnicas son utilizadas por las empresas de software en el Ecuador.

- Para hacer cálculos de esfuerzos, costos y tiempos en proyectos se debe tomar en cuenta la arquitectura y funcionalidad del sistema; por esta razón el uso de la herramienta MPE dará mejores resultados en la estimación inicial de los proyectos de software.

5.3 Ventajas Obtenidas De La Metodología MPE

- La nueva propuesta metodológica utiliza el modelo conceptual de datos inicial del sistema a desarrollar, ayudándonos a obtener resultados base, es decir, estimaciones iniciales que servirán de guía para conocer el límite inferior que necesita el sistema para ser desarrollado.

- MPE como herramienta, automatiza los aspectos importantes dentro del proceso de desarrollo de software, como son por ejemplo los mecanismos de acceso a datos, seguridad, comunicaciones, etc.

- El prototipo se implementa a través de una herramienta CASE, la misma que genera un conjunto de clases que aseguran el cumplimiento de las guías de arquitectura más comúnmente utilizadas en soluciones empresariales de alta disponibilidad requeridas actualmente por la industria.

- Para la generación del prototipo MPE se utiliza técnicas de programación de alta calidad que ayudan a tener una herramienta automática mucho más efectiva, que las pocas herramientas existentes en la actualidad.

6 CAPÍTULO VI - IMPLEMENTACIÓN DEL PRODUCTO

A continuación presentamos las características que ofrece MPE como prototipo de estimación.

Las siguientes Figuras muestran el aplicativo desarrollado para dar soporte a la automatización del método expuesto.

6.1 Generación del Prototipo

- Para la generación del Prototipo se utiliza la herramienta CodSmith Studio. Primero se realiza la selección del modelo de la base de datos, ingresando el string de conexión, para poder generar la arquitectura en base a ese modelo.

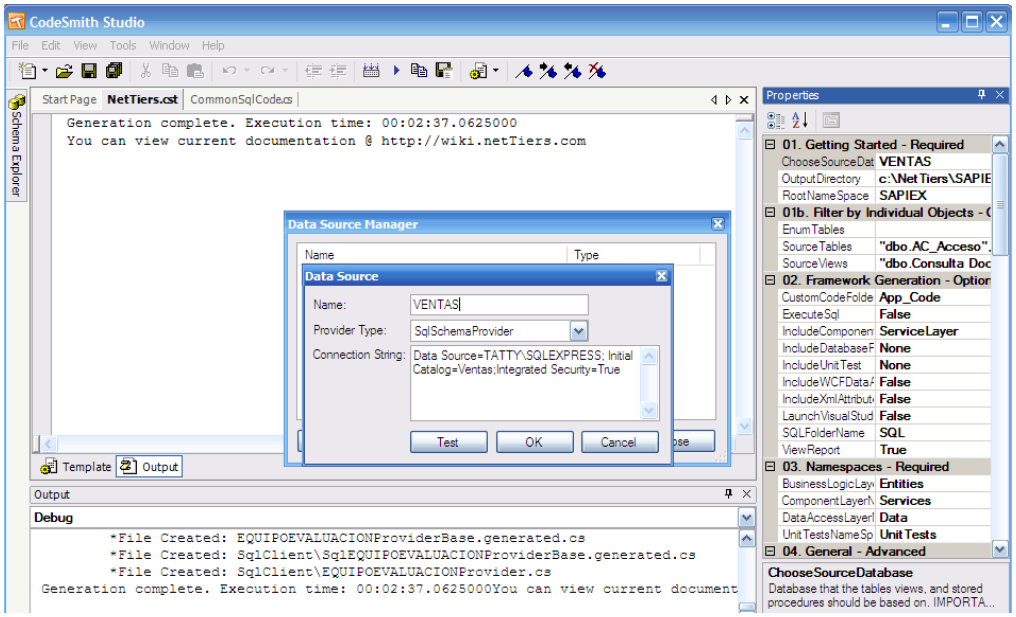


Figura 6.1 Selección del Modelo Conceptual

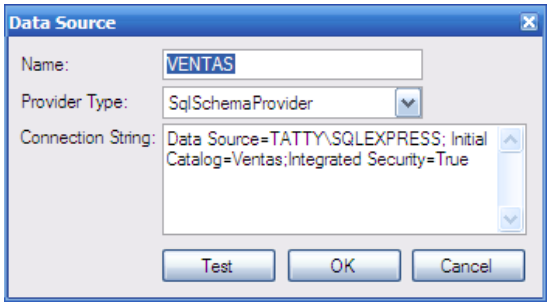


Figura 6.2 Configuración del String de Conexión

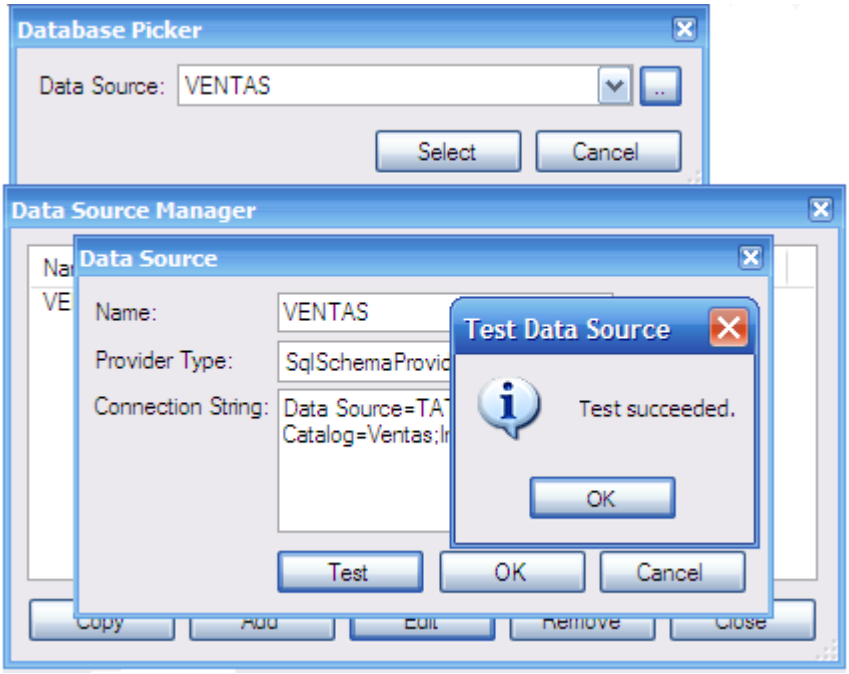


Figura 6.3 Prueba de Conexión con la Base de Datos

- Se selecciona el Modelo Conceptual y las tablas correspondientes para la generación de código por defecto de la arquitectura base del sistema. Para nuestro caso de estudio el Framework utiliza el lenguaje C#.

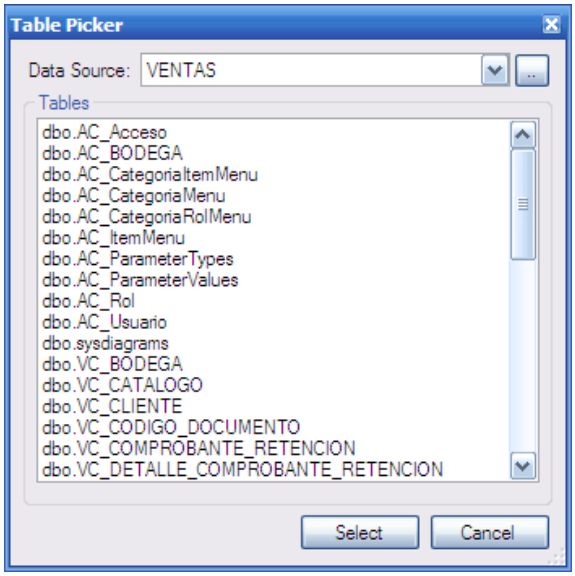


Figura 6.4 Selección de las tablas del Modelo Conceptual para generación de la Arquitectura

- Todos los ajustes necesarios se los realiza en las propiedades del sistema dentro de la herramienta CodeSmith y luego se ejecuta la acción para la generación completa del código base de la arquitectura, creándose dentro del proyecto un archivo *.sln;* este archivo generado ayudará al conteo de líneas de código dentro de la aplicación MPE.

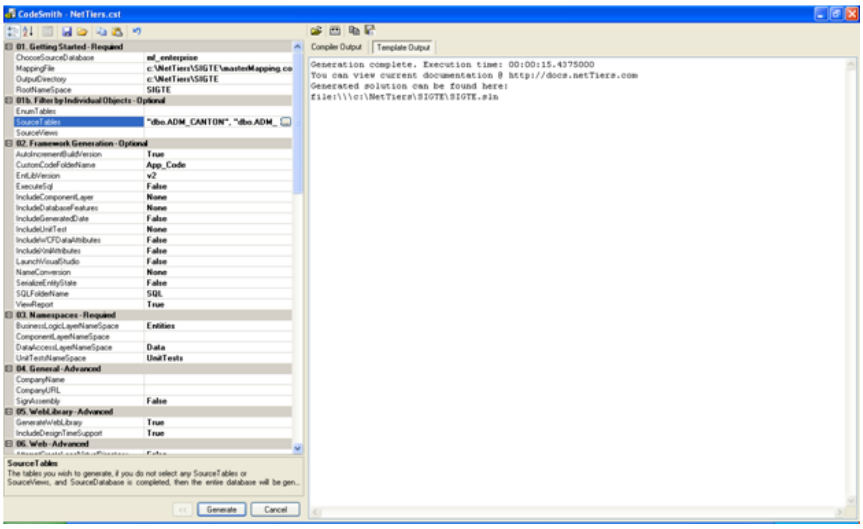


Figura 6.5 Configuración de Propiedades y generación de la arquitectura

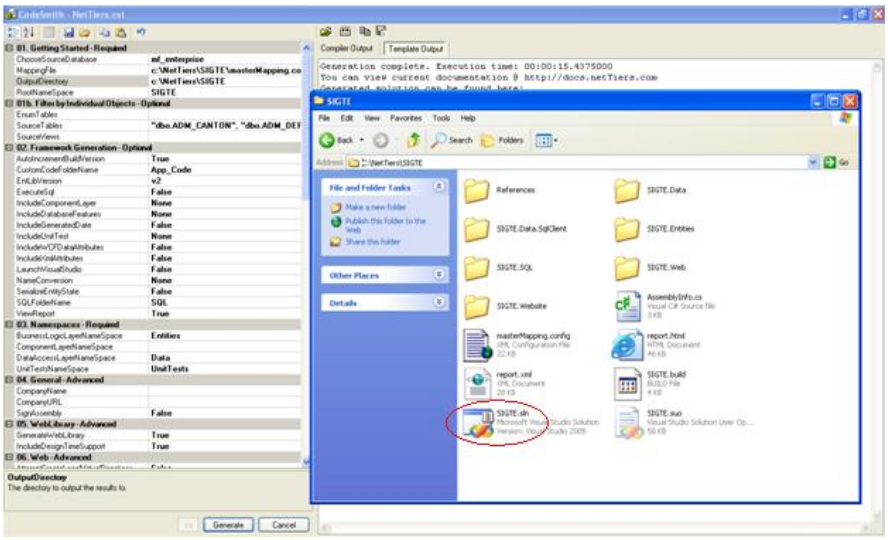


Figura 6.6 Creación y ubicación física del archivo *.sln*

- Dentro del software modelo MPE, se presenta la pantalla inicial y la pantalla de ingreso del modelo Entidad Relación, donde seleccionamos el archivo *.sln* que tiene la generación de la arquitectura del sistema, para convertirla en líneas de código.

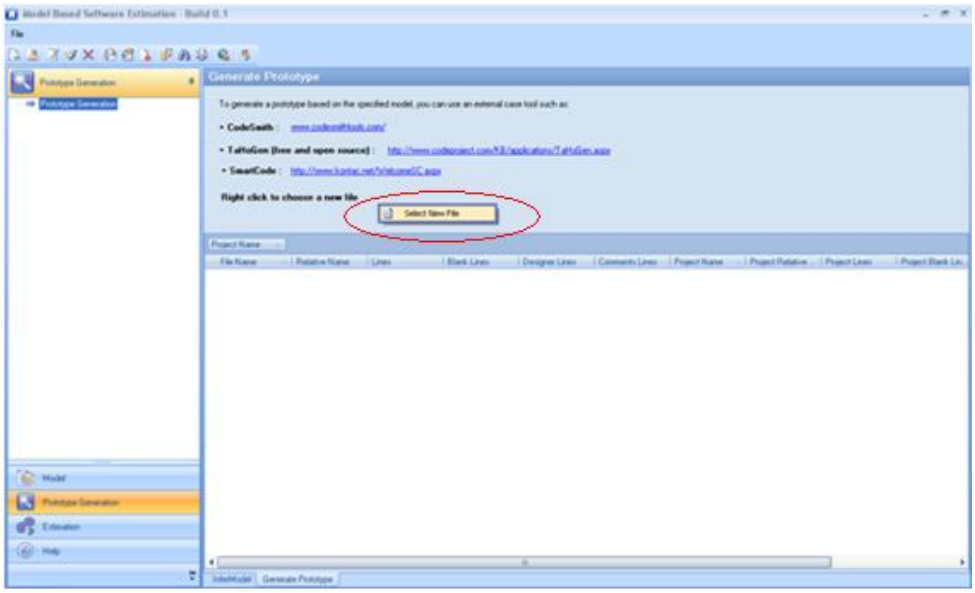


Figura 6.7 Selección del Archivo del Modelo Conceptual

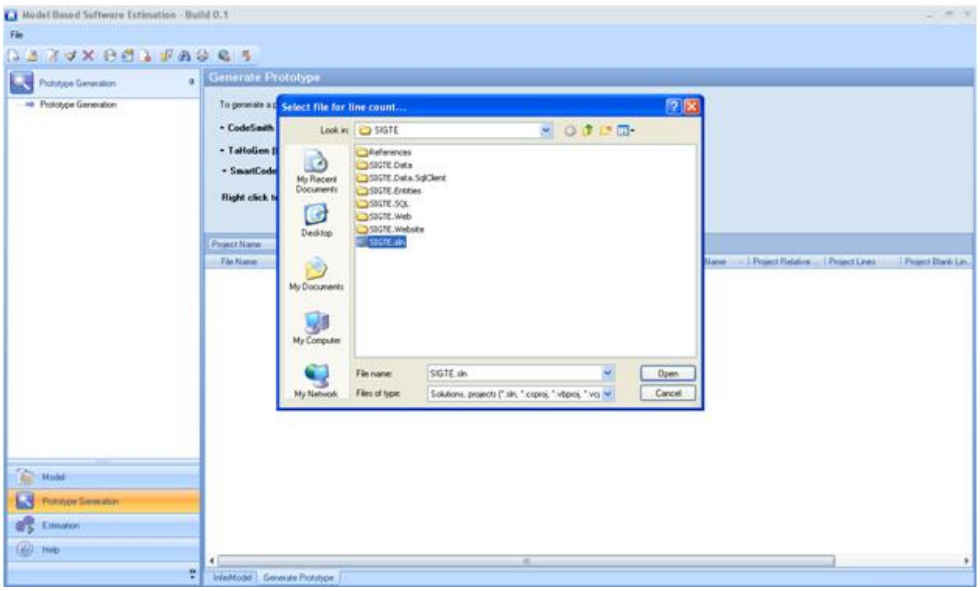


Figura 6.8 Selección del archivo generado por CodeSmith Studio

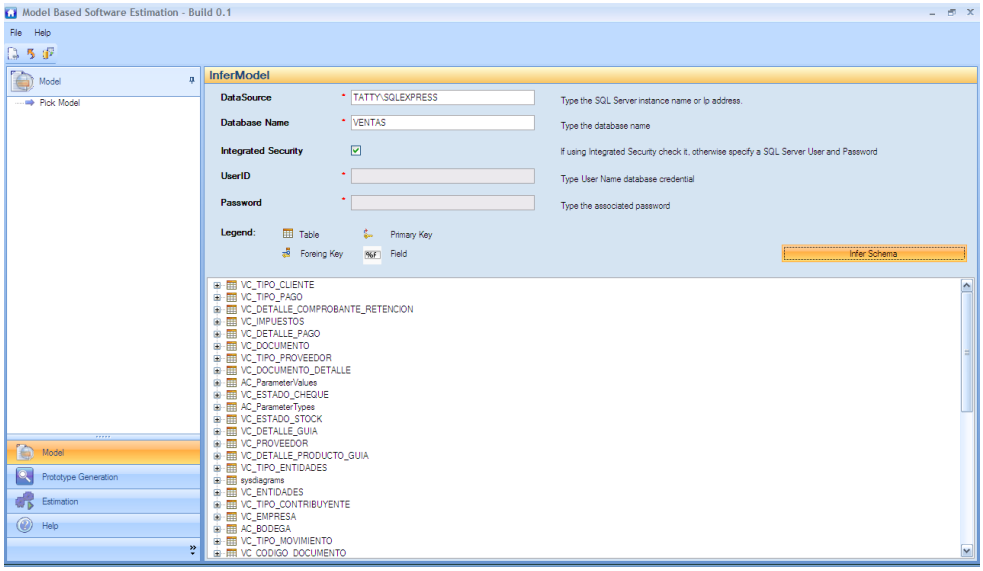


Figura 6.9 Descripción de tablas del Modelo Entidad Relación

6.2 Conteo del número de líneas de código del Prototipo

Las Figuras siguientes, muestran las pantallas del conteo de de número de líneas de código obtenidos por el software desarrollado.

- A continuación se realiza la descripción del total de líneas de código que tiene el sistema, el total de líneas comentadas, en blanco y archivos de código.

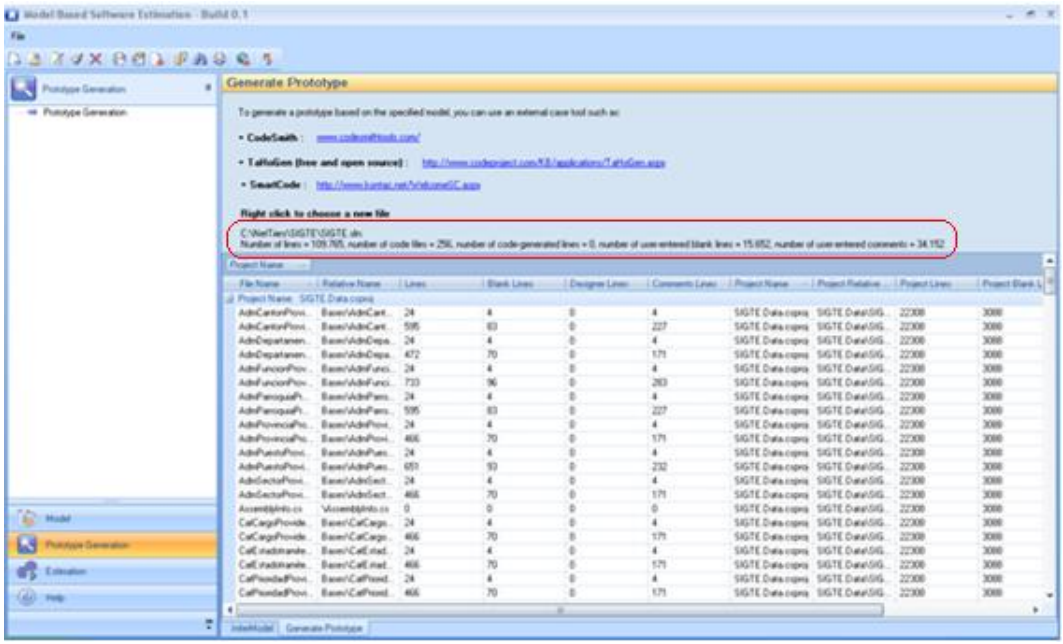
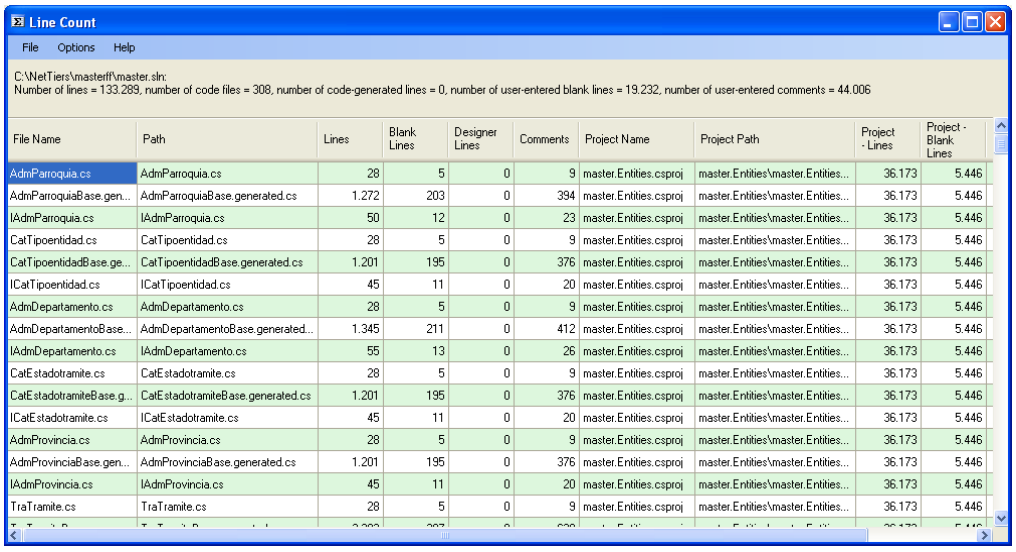


Figura 6.10 Descripción del Total de Líneas de Código Generadas



File Name	Path	Lines	Blank Lines	Designer Lines	Comments	Project Name	Project Path	Project - Lines	Project - Blank Lines
AdmParroquia.cs	AdmParroquia.cs	28	5	0	9	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
AdmParroquiaBase.gen...	AdmParroquiaBase.generated.cs	1.272	203	0	394	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
IAdmParroquia.cs	IAdmParroquia.cs	50	12	0	23	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
CatTipoentidad.cs	CatTipoentidad.cs	28	5	0	9	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
CatTipoentidadBase.ge...	CatTipoentidadBase.generated.cs	1.201	195	0	376	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
ICatTipoentidad.cs	ICatTipoentidad.cs	45	11	0	20	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
AdmDepartamento.cs	AdmDepartamento.cs	28	5	0	9	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
AdmDepartamentoBase...	AdmDepartamentoBase.generated...	1.345	211	0	412	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
IAdmDepartamento.cs	IAdmDepartamento.cs	55	13	0	26	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
CatEstadotramite.cs	CatEstadotramite.cs	28	5	0	9	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
CatEstadotramiteBase.g...	CatEstadotramiteBase.generated.cs	1.201	195	0	376	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
ICatEstadotramite.cs	ICatEstadotramite.cs	45	11	0	20	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
AdmProvincia.cs	AdmProvincia.cs	28	5	0	9	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
AdmProvinciaBase.gen...	AdmProvinciaBase.generated.cs	1.201	195	0	376	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
IAdmProvincia.cs	IAdmProvincia.cs	45	11	0	20	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446
TraTramite.cs	TraTramite.cs	28	5	0	9	master.Entities.csproj	master.Entities\master.Entities...	36.173	5.446

Figura 6.11 Conteo de líneas de código.

- El software de soporte desarrollado permite una validación empírica más eficaz del método; se ha conseguido determinar los indicadores necesarios en el menor tiempo y precisión posible.

7 CAPÍTULO VII - CONCLUSIONES Y RECOMENDACIONES

7.1 Conclusiones

- Con los resultados obtenidos por la aplicación de la nueva propuesta metodológica a proyectos reales de software, se puede confirmar la hipótesis planteada, esto es, que el esfuerzo necesario para desarrollar una Arquitectura de Software puede llegar a ser mayor que el desarrollo de la funcionalidad misma del sistema; aunque este planteamiento no puede generalizarse ya que se hace necesaria una validación del método.

- La investigación de campo realizada a las empresas desarrolladoras de software, ayudó para definir la situación actual y real sobre la estimación, ya que no tienen una técnica, ni metodología aplicable para el tipo de proyectos a desarrollar; por lo general se basan en la experiencia profesional del equipo de trabajo. Por esta razón la falencia en estimaciones tempranas produce desfases en la entrega de proyectos.

- La especificación de requerimientos no es considerada como etapa dentro del desarrollo de proyectos de software, y se utiliza erradamente el modelo conceptual de datos como parte de la etapa de diseño.

- El método Model Prototype Estimation (MPE) apoya en la medición temprana y logra definir el esfuerzo, tiempo y recurso humano necesarios para cumplir con los requisitos funcionales y no funcionales, específicamente de diseño arquitectónico.

- El método puede criticarse en el sentido de que está asociado a una tecnología en particular y en consecuencia no podría estandarizarse, sin embargo, esta

deficiencia se compensa cuando se consigue en etapas tempranas de desarrollo obtener una estimación real, con un prototipo funcional que puede ser validado por el cliente, previo al desarrollo del producto final.

- Para realizar el cálculo de los puntos de función debe existir un consenso dentro de la organización para el tratamiento de los requisitos, además de realizar un pre diseño para que este tamaño sea una buena estimación del tamaño real de la aplicación.

- Se requiere prototipar la arquitectura para conseguir: mejor definición de requisitos, mejor ejecución del plan y lograr calidad del producto.

- Se ha comprobado que se puede definir la arquitectura del sistema antes del desarrollo de todo el software, por esta razón es posible realizar la estimación funcional y de arquitectura de proyectos.

- Con la metodología se puede desarrollar el sistema en menor tiempo, definiendo la arquitectura y trabajando en paralelo con los requisitos funcionales.

- El proyecto de metodología está apoyando la etapa 6 de la arquitectura de desarrollo de software: **Planificación de la Gestión de Proyectos,** porque define funcionalidad como arquitectura.

- Con la utilización del prototipo MPE, las empresas desarrolladoras de software tendrán la posibilidad de entregar un producto de alta calidad, con resultados más reales al momento de realizar la estimación.

7.2 Recomendaciones

- Una buena estimación se realiza siguiendo las etapas de visionamiento, análisis, planeamiento, etc. de desarrollo de software, pero actualmente las empresas de software lo realizan según su conveniencia; por esta razón es bueno mantener un esquema para realizar una estimación real del sistema.

- Es necesario hacer énfasis en que el método tiene su aplicación específica para sistemas que privilegian los datos a los procesos, mas no para otro tipo de software como pueden ser sistemas en tiempo real o sistemas con arquitecturas orientadas a servicios que involucran la integración de muchas tecnologías y sistemas legados.

- Una vez adoptado el método se consigue como beneficio adicional que el framework desarrollado puede ser reutilizable, dando la oportunidad de ir recopilando datos de manera automática propendiendo a que las futuras decisiones de planeación puedan realizarse con mayor grado de certeza, que es lo que finalmente buscan los gerentes de sistemas y sus clientes.

- Actualmente en el Ecuador, existen proyectos que son orientados a prestar servicios, esto implica mayor desarrollo en la arquitectura del sistema, superando el peso asignado a la funcionalidad; pero a pesar de esto, no se le da un valor representativo a la arquitectura el momento de estimar los proyectos, ya que los métodos de estimación existentes se basan simplemente en la funcionalidad del sistema. Es por esta razón, que se recomienda utilizar el método de estimación desarrollado, porque considera importante tanto la funcionalidad como la arquitectura del sistema, buscando el equilibrio en la estimación de los sistemas de software.

- Se recomienda publicar los resultados obtenidos y difundir la nueva propuesta metodológica para estimación de software.

- Se recomienda que el prototipo MPE sea aplicado a los proyectos de tesis de la Carrera de Ingeniería en Sistemas e Informática de la ESPE, y que los resultados obtenidos sean usados para la comprobación de la metodología.

7.3 Referencias

- Allan J. Albrecht and John E. Gaffney, "Software Function, Lines of Code, and Development Effort Prediction: A Software Science Validation", IEEE Transactions on Software Engineering, vol SE-9, No 6, November 1983.

- Jack E. Matson, Bruce E. Barrett, and Joseph M. Mellichamp, "Software Development Cost Estimation using Function Points", IEEE Transactions on Software Engineering, vol 20, no 4, April 1994.

- Barry W. Boehm and Philip N. Papaccio, "Understanding and Controlling Software Costs", IEEE Transactions on Software Engineering, vol. 14, no 10, October 1988.

- Boehm, B. and W. Royce, *Ada COCOMO and the Ada Process Model*, Proceedings, Fifth COCOMO Users' Group Meeting,

- Graham C. Low and D. Ross Jeffery, "Function Points in the Estimation and Evaluation of the Software Process", IEEE Transactions on Software Engineering, vol 16, no 1, Juanary 1990.

- Charles R. Symons, "Function point Analysis: Difficulties and Improvements", IEEE Transactions on Software Engineering, vol 14, no1, January 1988.

- Jeffery, G. C. Low, and M. Barnes, "A Comparison of Function Point Counting Techniques", Transactions on Software Engineering, vol 19, no 5, May 1993.

- Norman Fenton, "Software measurement: A Necessary Scientific basis", IEEE Transactions on Software Engineering, vol 20, no 3, March 1994.

- Tridas Mukhopadhyay and Sunder Kekre, "Software effort Models for Early Estimation of Process Control Applications", Transactions on Software Engineering, vol 18, no 10, october 1992.

- W. Boehm, "Software Engineering Economics", Englewood Cliffs, NJ: Prentice - Hall, 1981.

- Roger S. Pressman, "Ingeniería del Software, Un Enfoque Practico", tercera Edición, McGraw-Hill editores, 1994.

- Ian Sommerville, "Software Engineering", Fourth Edition, Addison- Wesley Publishing Company, 1992.

- Caper Jones, "Programming Productivity", McGraw-Hill, 1986

- Marcela Varas C, "Modelo de Gestión de Proyectos Software: Estimación del Esfuerzo de Desarrollo", Encuentro Chileno de Computación, Arica 1995.

- Allan J. Albrecht and John E. Gaffney, November 1983. "Software Function, Lines of Code, and Development Effort Prediction: A Software Science Validation", IEEE Transactions on Software Engineering, vol SE-9, No 6.

- W. Boehm. 1981. "Software Engineering Economics", Englewood Cliffs, NJ: Prentice-Hall.

- Losavio F. Diciembre 2006. “Marco conceptual para un diseño arquitectónico basado en aspectos de calidad”, Revista Universitaria de Investigación SAPIENS, vol 7, No 2.

- Krutchen P. 2000. The Rational Unified Process. An Introduction. Second Edition. Addison-Wesley. Readings. Massachusetts.

- Bosh, J. 2000. Design & Use of Software Architectures: Adopting and evolving a product-line approach. Great Britain: Pearson Education Limited.

- Kazman, R., Klein, M., Barbacci, T., Longstaff, H., Lipson, H., y Carriere J. 1998. The Architecture Tradeoff Analyisis Method. IEEE, ICECCS.

- IEEE Std 1471-2000, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems, IEEE, 2000.

- IEEE (1991). IEEE Std. 1074-1995, IEEE Standard for Developing Software Life Cycle Processes Nueva York: The Institute of Electrical and Electronics Engineers, Inc.

- Casanovas J. julio 2004. “Usabilidad y Arquitectura del Software”, Disponible en: http://www.alzado.org/articulo.php?id_art=355.

- David Garlan and Mary Shaw. New Jersey, 1993. "An Introduction to Software Architecture", Advances in Software Engineering and Knowledge Engineering, Volume I, edited by V.Ambriola and G.Tortora, World Scientific Publishing Company.

- Fenton, N. E. y Pfleeger, S. L. 1997. Software Metrics. A Rigorous & Practical Approach. International Thomson Computer Press.

- Graham, Henderson-Sellers y Younessi (1997) definen el término en su libro "The OPEN Process Specification".

- Briand, L. C., Morasca, S. y Basili, V.R. (1996). Property-Based Software Engineering Measurement. *IEEE Transactions on Software Engineering*, 22(1).

- McTaggart, R. (1991). Principles of Participatory Action Research. *Adult Education Quarterly*, 41(3).

- Wadswroth, Y. (1998). What is Participatory Action Research? *Action Research International*. Accedido el 10 de enero de 2000 en Internet: http://www.scu.edu.au/schools/sawd/ari/ari-wadsworth.html

- French, W.L. y Bell, C.H. Jr. (1996). *Desarrollo organizacional (quinta edición)*. Naucalpán de Juárez, México: Prentice-Hall.

7.3.1 Páginas Web

- http://www.sc.ehu.es/jiwdocoj/mmis/fpmkii.htm
- http://www.sc.ehu.es/jiwdocoj/mmis/fpa.htm
- http://kybele.escet.urjc.es/documentos/HC/HC4GL2007-T4-MetricaV3-Tecnicas-Compl.pdf
- http://www.cs.us.es/cursos/bd-2001/temas/diseno.html
- http://www.inf.udec.cl/revista/ediciones/edicion1/mvaras.PDF
- http://es.wikipedia.org/wiki/M%C3%A9trica_de_punto_funci%C3%B3n
- http://www.willydev.net/descargas/willydev_planeasoftware.pdf
- http://sisbib.unmsm.edu.pe/BibVirtualData/publicaciones/risi/N1_2004/a10.pdf
- http://msdn.microsoft.com/en-us/library/bb421527.aspx
- http://www.codeproject.com/KB/architecture/usecasep.aspx
- http://www.stsc.hill.af.mil/crosstalk/2000/04/stutzke.html
- http://sunset.usc.edu/events/2000/oct24-27-00/Presentations/paulish.pdf

- http://www.icmgworld.com/corp/news/Articles/Tom/To14.asp
- http://delivery.acm.org/10.1145/250000/243625/p126-paulish.pdf?key1=243625&key2=0145466511&coll=&dl=ACM&CFID=15151515&CFTOKEN=6184618
- http://www.taringa.net/posts/downloads/833988/CodeSmith-Profesional-4_0-+-Crack.html

ANEXO A

Investigación De Campo De La Realidad Del Desarrollo De Software En El Ecuador

MODELO DE ENTREVISTA

ESTIMACIÓN DEL SOFTWARE

EMPRESA:

NOMBRE:

EDAD:

CARGO / PUESTO DE TRABAJO:

AÑOS DE EXPERIENCIA EN DESARROLLO DE SOFTWARE:

AÑOS DE ANTIGÜEDAD DE LA EMPRESA:

1. ¿Qué tipo de proyectos de software desarrolla dentro de la empresa? (De Gestión, Especializados). Indique, el tiempo, y número de personas empleado.

2. ¿Qué metodología(s) de desarrollo de software utiliza en los proyectos software dentro de la empresa? (Estructuradas, Orientadas a Objetos).

3. En base a dichas metodologías, ¿en qué fases divide el desarrollo de sus proyectos (Planificación, Análisis, Diseño, etc.)?

4. ¿Qué tipo de técnicas de modelado del sistema y su entorno utiliza generalmente (Diagrama de clase, diagrama de secuencia, Modelo Entidad-Relación, Diagrama de Flujo de Datos, etc.)?

5. ¿Utiliza alguna de estas técnicas de modelado para obtener o mejorar la descripción de los requisitos?

6. En cuanto a las herramientas CASE, ¿utilizan alguna para facilitar el trabajo? ¿Qué herramientas (Visio, Rational Rose, Dome, MetaEdit, ArgoUML (open source), etc.)?

7. ¿Qué estándares utiliza la empresa para asegurar la calidad en el desarrollo de los proyectos de software? ¿Tiene alguna certificación de calidad en los proyectos software (CMMI, SPICE, ISO/IEC, ISO 9001:2000, etc.)?

8. ¿Se cumplió con los plazos previstos en la estimación inicial del proyecto de software?

9. ¿Qué tipo de métricas utiliza para el desarrollo de los proyectos software (De Calidad, Productividad, Orientadas a la Persona, Orientadas al Tamaño, Orientadas a la Función)?

10. ¿Qué tipo(s) de técnicas de estimación realiza para Planificar el desarrollo del proyecto de software (Ej. Datos Históricos, Técnicas Delphi, Líneas de Código y Puntos de Función, Puntos Objeto)?

11. ¿Cómo se determina el tiempo, costo y RRHH estimado necesario para el desarrollo del proyecto de software en la empresa?

12. ¿A la hora de estimar un proyecto de software, toma como base alguno de los modelos conceptuales descritos anteriormente (Modelo Entidad-Relación, Diagrama de Clase, Diagrama de Secuencia, etc.)?

13. ¿En base a qué parámetros distribuye el tiempo para cada una de las fases? ¿Tiene alguna proporción? (Análisis, diseño, implementación, pruebas).

14. ¿Dedica más tiempo del desarrollo del software a la funcionalidad o a la arquitectura?

15. ¿Conoce Ud. de algún sistema automático de predicción de tiempo, costo y RRHH para el desarrollo de los proyectos de software? Si la respuesta es afirmativa, ¿sabe en qué técnica se basa para el análisis?

16. Una buena estimación supone uno de los aspectos claves para la correcta planificación de un proyecto, ¿Cómo determinaría mejoras en el proceso de estimación a utilizar dentro de cualquier proyecto de software?

RESULTADOS OBTENIDOS DE LA INVESTIGACIÓN

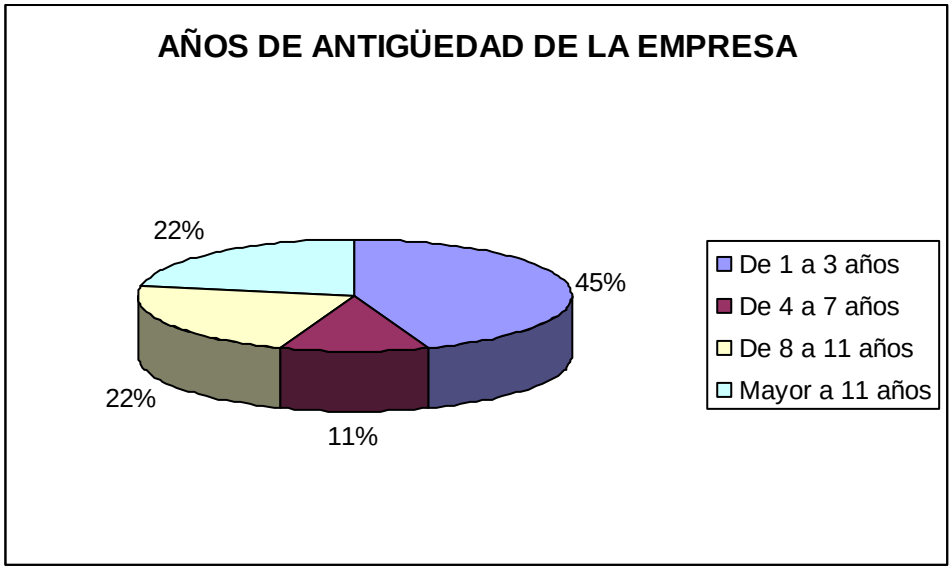
A continuación se presenta la realidad actual de las empresas de software en el Ecuador. El sector de mercado al que está orientado son las empresas desarrolladoras de software que realicen proyectos de Gestión o Especializados.

Como base de muestreo para el análisis de mercado se ha considerado varias empresas de las ciudades principales del país Guayaquil y Quito:

- Intergrupo
- Creative Works
- Extremo Software
- Infoelect
- Kruger Corporation
- Logicstudio
- Macosa
- Sdconsult
- Vimeworks
- Represensa Consulting Group
- Eikon S.A.
- OMTech
- Megatelcon S.A.

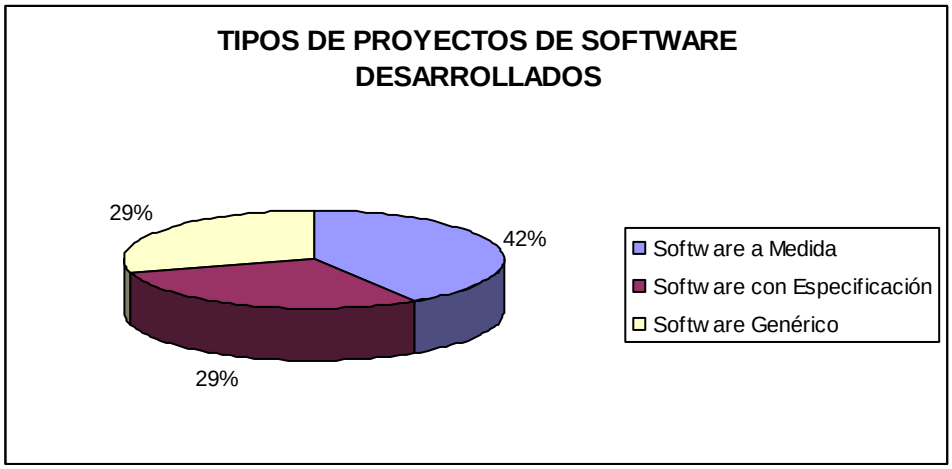
- Corporación Latinoamericana De Software
- Agrosoft S.A.

A)



El desarrollo de software en el Ecuador es una industria nueva con rápido crecimiento, pero que requiere refinar y establecer normas y estándares para un mejor desempeño que asegure el cumplimiento de objetivos.

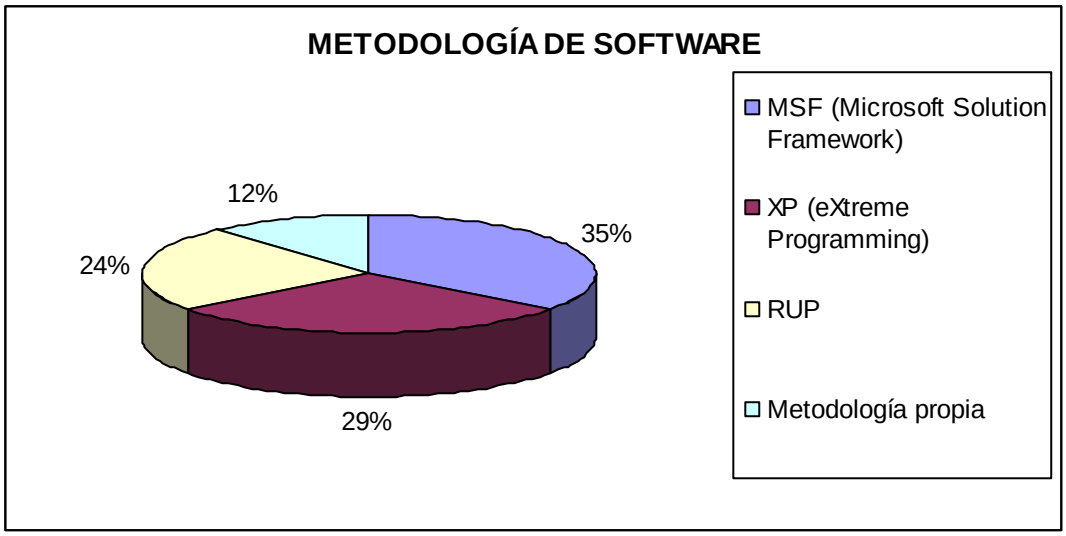
B)



La mayoría de empresas de software del país desarrollan proyectos de software a la medida, de acuerdo a la especificación de requerimientos del cliente.

Se observa también, que existen algunas áreas de desarrollo de sistemas software que no se explotan en el Ecuador, como son: Sistemas de Automatización y Control, Sistemas de Inteligencia Artificial, Sistemas Empotrados, etc.

C)



MSF[1] es la metodología más utilizada por las empresas desarrolladoras de software a nivel nacional, ya que les garantiza soporte, certificaciones y técnicas de fácil implementación.

Microsoft ha monopolizado las pequeñas empresas de desarrollo en el Ecuador.

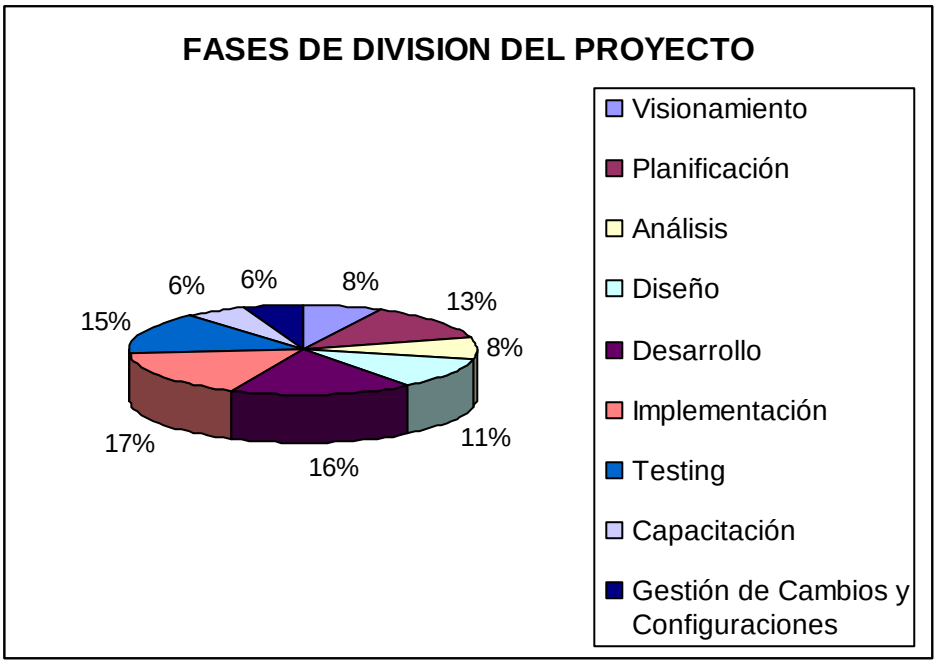
En la actualidad se utiliza la metodología RUP[2], porque realiza el ciclo de vida del software de forma interactiva, reemplazando a la metodología en Cascada que se convierte en una forma tediosa y larga para desarrollar.

[1] Microsoft Solution Framework

[2] (El **Proceso Unificado de Rational** (*Rational Unified Process*) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para

Las últimas tendencias para el desarrollo de proyectos de software, está dividida de forma equitativa entre la metodología XP[1] y la metodología de proyectos PMI[2], ya que XP se basa en la especificación de requisitos para la satisfacción del cliente del software a desarrollar; mientras PMI asigna mayor tiempo a la fase de planificación dentro de los proyectos de software a desarrollar.

D)



En la actualidad dentro del ciclo de vida de los proyectos de software no se le da la importancia necesaria a la fase de Planificación y Análisis, sino se centran en el Desarrollo e Implementación del software; por este motivo existen falencias en el proceso de estimación de proyectos, en cuanto a los tiempos, costos y recurso humano asignado.

---

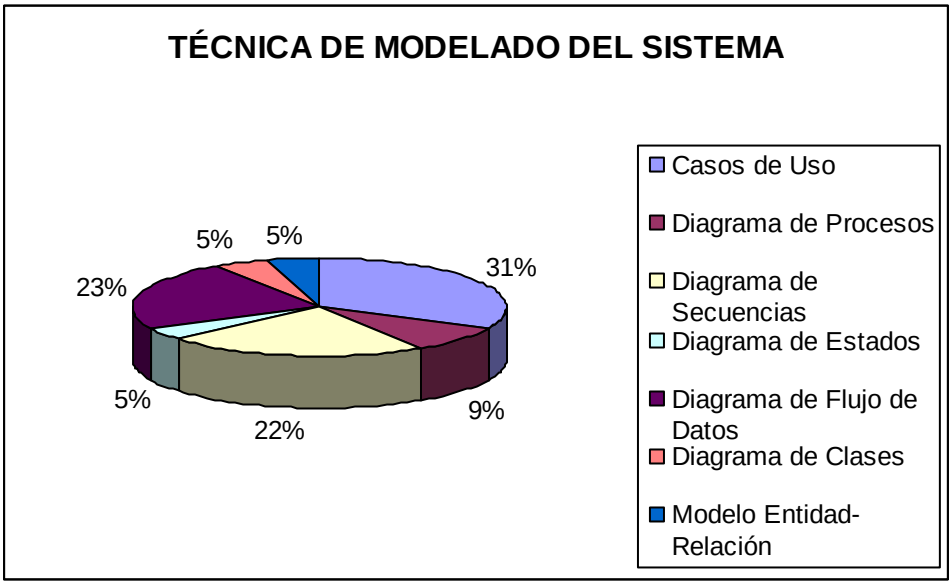
el análisis, implementación y documentación de sistemas orientados a objetos - http://es.wikipedia.org/wiki/Proceso_Unificado_de_Rational)

[1] (La **programación extrema** o *eXtreme Programming* (XP) es un enfoque de la ingeniería de software formulado por Kent Beck. La programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad - http://es.wikipedia.org/wiki/Programaci%C3%B3n_Extrema)

[2] (Certificación Project Management Institute)

Muy pocas empresas de software en el Ecuador considera la fase de Capacitación como importante dentro del ciclo de vida de los proyectos. Por esta razón, así el software este bien desarrollado; el cliente no encontrará satisfacción en el mismo, lo cual generará pérdidas en tiempo, costos y personal para ambas partes.

E)



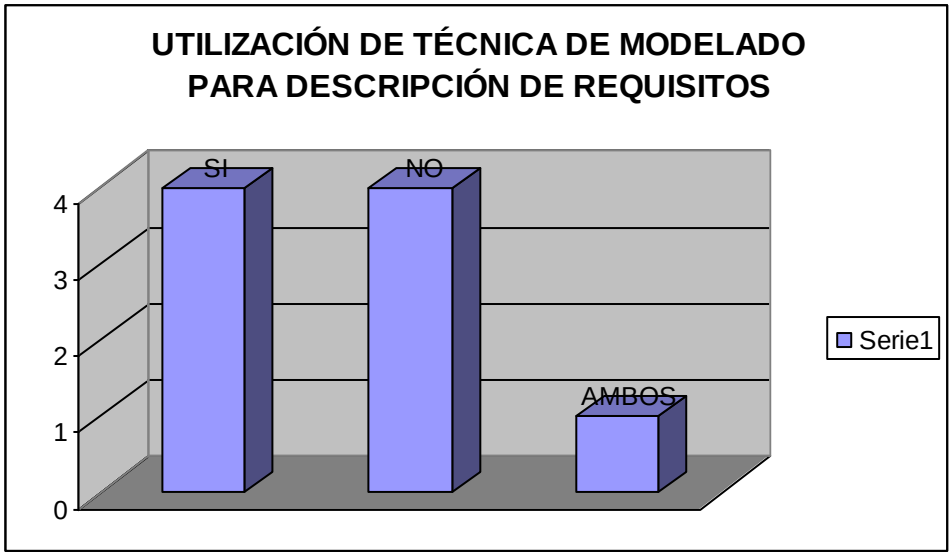
La mayoría de empresas desarrolladoras de software emplean como técnica de modelado Casos de Uso para el desarrollo del sistema y su entorno, ya que es una forma rápida y clara para la definición de requisitos y necesidades del cliente.

Existen diagramas UML[1] que no son considerados como importantes, ya que es una pérdida para las empresas dedicar mucho tiempo a la fase de Planificación diseñando diagramas; realizándolos básicamente cuando los procesos del proyecto de software son muy complejos.

[1] (Lenguaje Unificado de Modelado, Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. - http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado)

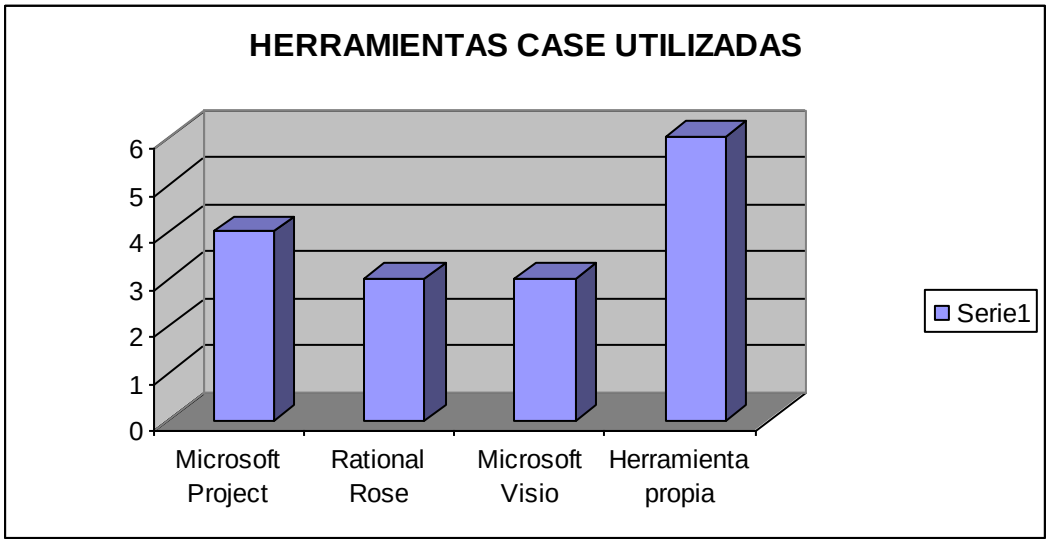
El Modelo Entidad-Relación es uno de los diagramas menos considerados dentro de la técnica de modelado de proyectos de software, ya que se lo realizará en la fase de Desarrollo, añadiendo clases y atributos de acuerdo a la necesidad actual del sistema.

F)



La utilización de una técnica de modelado para la descripción de requisitos esta dividida, ya que existen empresas que no consideran imprescindible su uso, ya que emplean la especificación de requerimientos inicial. Mientras que otras empresas consideran que si es necesario para que los requisitos sean mejor definidos y aplicados al momento del desarrollo del proyecto de software. Las pocas empresas que consideran que algunas veces se debe utilizar la técnica y otras veces no, se debe a la complejidad y tamaño del proyecto de software a desarrollar.

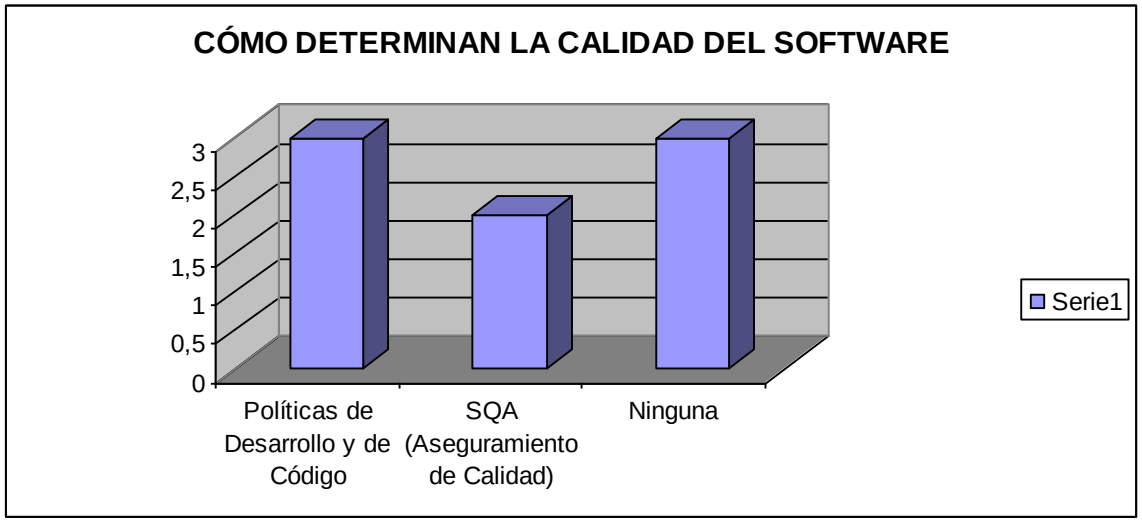
G)

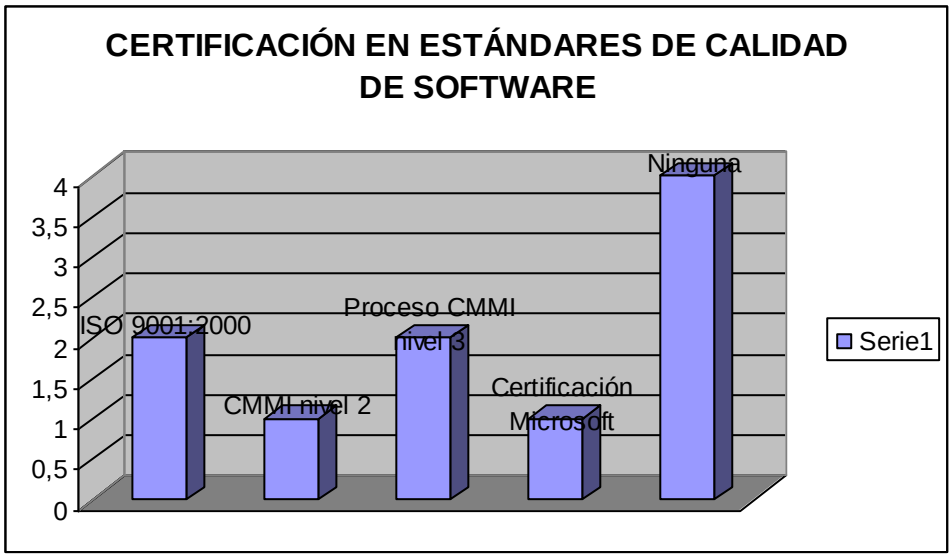


La mayoría de empresas desarrolladoras de software consideran que es mejor utilizar una herramienta propia diseñada de acuerdo a las necesidades de la empresa, que faciliten el desarrollo de la planificación dentro de cada proyecto.

Las herramientas Microsoft son las más utilizadas por las empresas desarrolladoras de software, que tienen poco tiempo de posicionamiento en el mercado, y recursos limitados.

H)

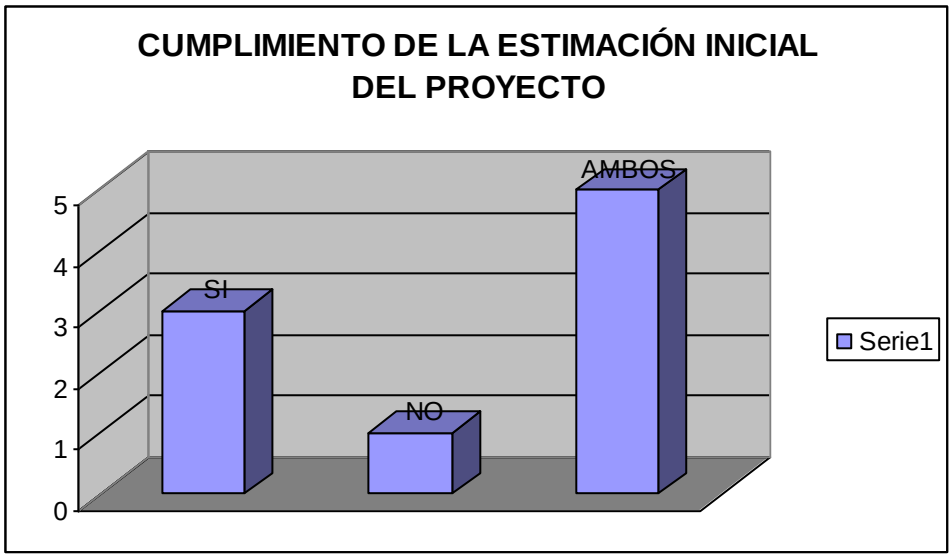




Muy pocas empresas de software tienen interés en obtener una certificación de calidad de software, ya que el personal deberá estar dedicado a tiempo completo para el desarrollo de cada fase en la obtención de una certificación, lo cual genera grandes pérdidas para la empresa.

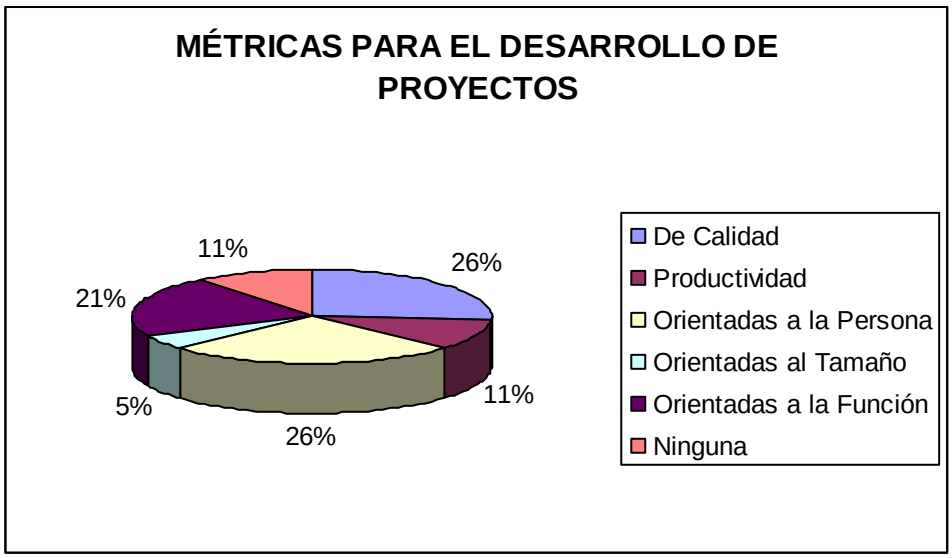
Los estándares de software, están principalmente enfocados en la reutilización de código y empleo de módulos previamente establecidos y aprobados; y solo un pequeño porcentaje de empresas utilizan estándares enfocados en la calidad en el desarrollo de proyectos.

I)



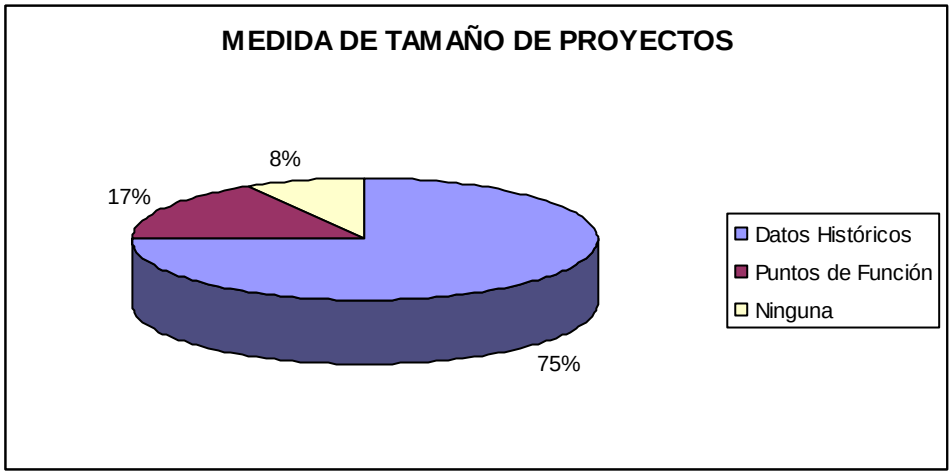
La variación en el cumplimiento de la estimación inicial de proyectos se debe principalmente a dos factores; primero la insatisfacción del cliente y por esta razón se debe reestimar proyectos continuamente, y fallas en la asignación de recursos, tiempo y costos para cada proyecto. Es decir, no existe un análisis de la gestión de riesgos adecuado.

J)



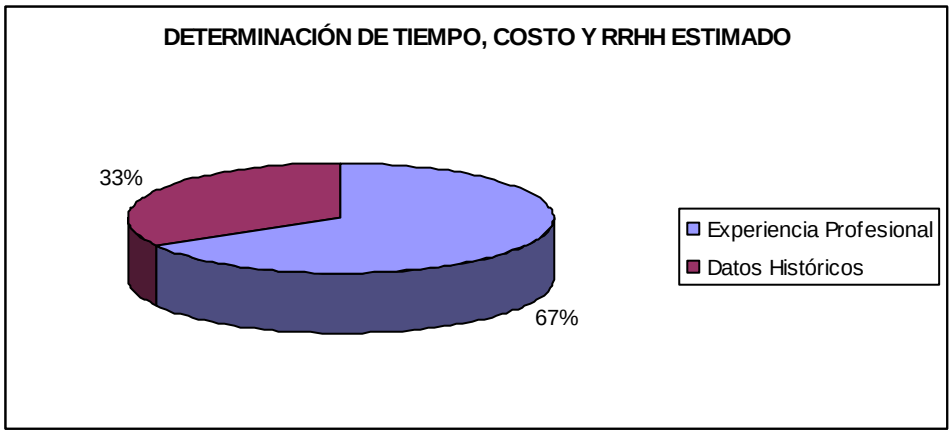
Las métricas de Calidad y Orientadas a la Persona son las más utilizadas dentro del desarrollo de proyectos, ya que son los factores medulares para un cumplimiento satisfactorio de proyectos.

K)



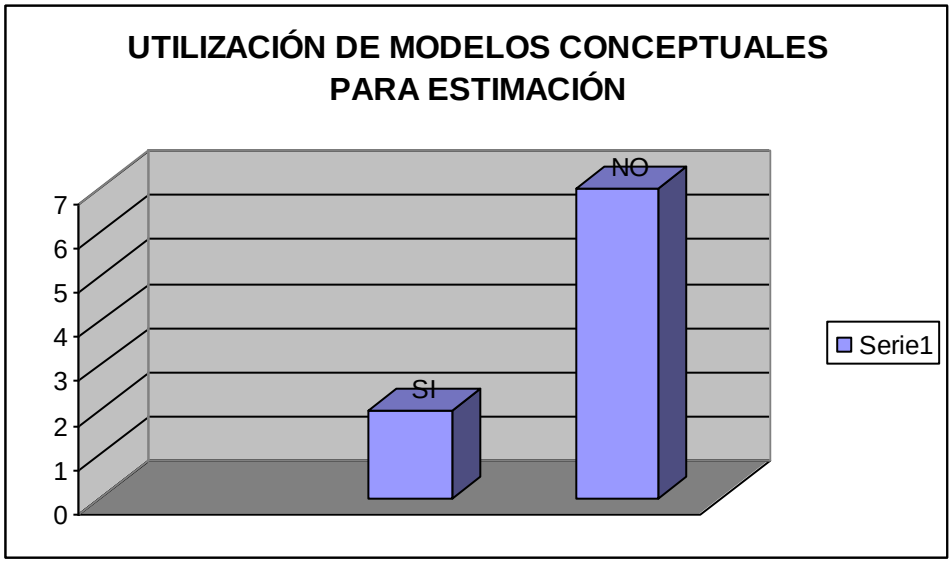
Las empresas ecuatorianas de desarrollo software en su totalidad utilizan datos históricos para realizar la estimación de proyectos, ya que son apegados a la realidad, sin desperdiciar mucho tiempo y responden a la experiencia profesional de su recurso humano.

L)



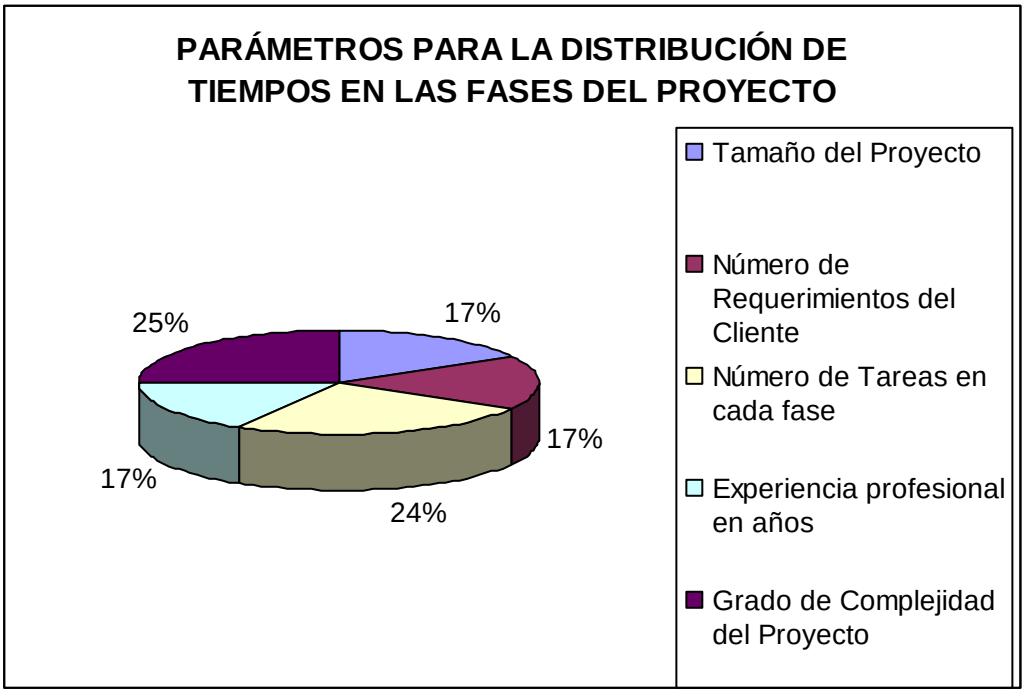
Para la determinación de tiempo, costo y recurso humano dentro de la planificación de proyectos, se considera la experiencia y capacitación profesional como primera opinión al momento de decidir y asignar recursos al desarrollo de proyectos, sin considerar los márgenes de riesgos y errores que se pueden presentar en desarrollo de proyectos.

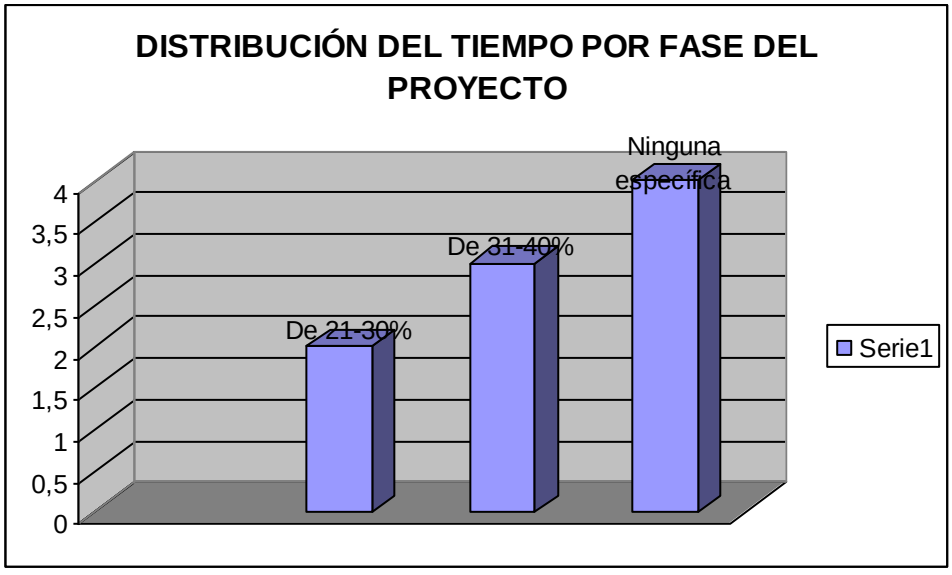
M)



La mayoría de empresas desarrolladores no utiliza los modelos conceptuales como base para una buena estimación de proyectos, ya que ha sido reemplazado por Datos Históricos y Experiencia profesional principalmente.

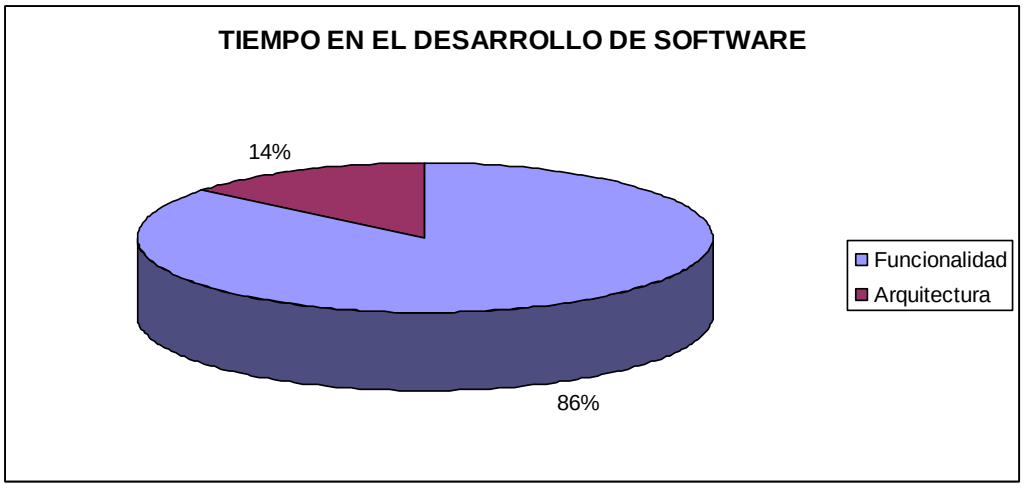
N)





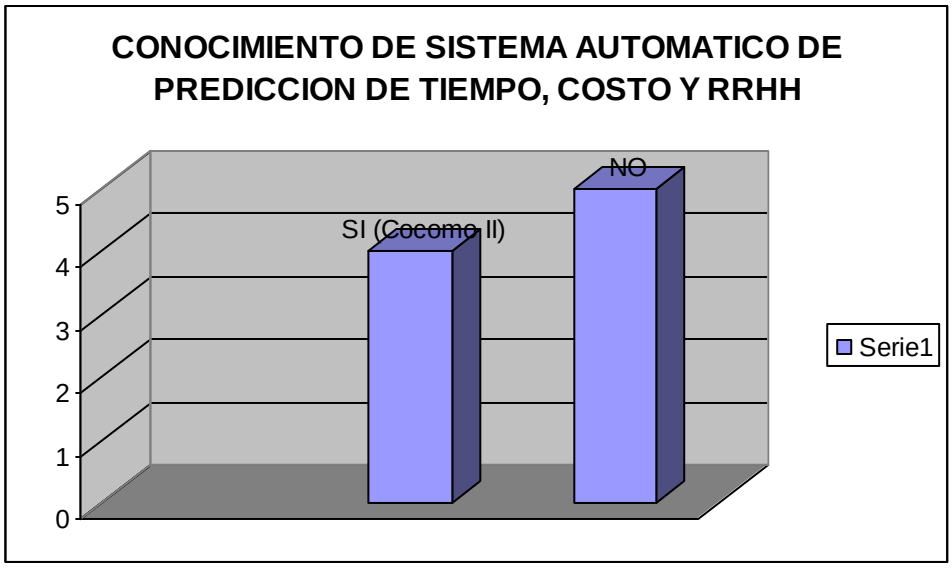
Se considera la complejidad del proyecto como parámetro importante en la distribución de tiempo para cada fase de desarrollo, ya que guarda una proporcionalidad directa entre el tiempo y las tareas asignadas para cada fase.

O)



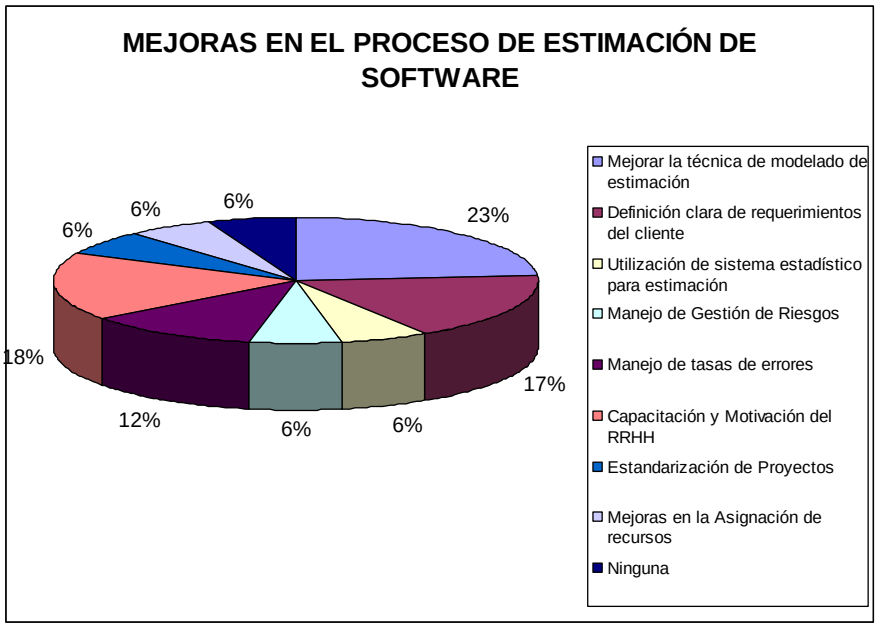
Las empresas de software en nuestro país dedican más tiempo a la funcionalidad en el desarrollo de software, ya que consideran que la lógica de negocio es lo más importante, para satisfacción del cliente al momento de entregar el producto final, con mayor calidad.

P)



La mayoría de empresas desarrolladoras no conocen de ningún sistema automático de estimación; cabe señalar que el porcentaje afirmativo solo conoce a Cocomo II como una herramienta de números fríos que no se apegan a la realidad, por ello su respuesta se limita a una conceptualización del mismo.

Q)



Se observa que no existe ninguna técnica o herramienta de estimación de software utilizada en los proyectos a desarrollar, ya que existe falta de conocimiento, no se apegan a la realidad y poseen altos índices de error, por lo cual optan en planificar sus proyectos de acuerdo a la experiencia y datos históricos que poseen.

Debe existir una mejor capacitación y motivación del recurso humano al momento de desarrollar nuevos proyectos, ya que el factor más problemático es el trabajo en equipo que produce inconsistencias entre las fases del proyecto, lo cual refleja pérdidas en tiempo y costos.

Se debe recalcar en las cláusulas de los contratos de proyectos de forma explicita, lo que ofrece el sistema de acuerdo a la demanda del cliente, evitando dejar vacíos que a la larga generen desacuerdos e inconformismo entre las partes.

ANEXO B
Sistema De Gestión De Trámites Del Estado SIGTE

ESPECIFICACION DE REQUISITOS DE SOFTWARE

Ver. 1.0

II.1. Introducción

En este apartado se detallará un modelo de Especificación de Requisitos Software (ERS) para un sistema ejemplo de desarrollo de software. Esta especificación toma como base el estándar "IEEE Recommended Practices for Requirements Specification ANSI/IEEE st. 830, 1998.", y el mismo puede ser ampliado o mejorado por el alumno.

II.1.1. Propósito

Este documento, presenta la especificación de requisitos de software (ERS) del Sistema de e-Government para registro y control de trámites mediante el uso de una herramienta de desarrollo rápido de aplicaciones (RAD).

Esta especificación se basa en el estándar "IEEE Recommended Practices for Requirements Specification ANSI/IEEE st. 830, 1998". Pretende abstraer principalmente los conceptos funcionales del producto que se espera realizar, y está dirigido especialmente al grupo de desarrollo, grupo de calidad y usuario final.

Las revisiones que se hagan a este documento, serán debidamente registradas tomando como base el Plan de Gestión de la Configuración definido para este propósito.

II.1.2. Ámbito y Alcance

El presente proyecto tiene como objetivo principal realizar una aplicación de e-Government para registro y control de trámites oficiales, esto es, los pedidos de atención a un requerimiento particular por parte de una entidad u organismo público o privado hacia el Estado, para lo cual se definirán los procesos y entes que intervienen en esta

actividad esperando mantener en todo momento un control eficaz del trámite así como de la documentación en formato digital que se vaya generando desde su ingreso al organismo de Gobierno, estudio por parte del departamento competente, hasta su resolución final y archivo respectivo.

El sistema futuro se identificará como "SIGTE - Sistema de Gestión de Trámites del Estado".

II.1.3 Definiciones, acrónimos y abreviaturas.

Definiciones:

Entidad	Organismo o Institución pública o privada legalmente constituida
Trámites Oficiales	Hace referencia a todo tipo de pedidos que genere una Institución, pública o privada y que lleven una firma de responsabilidad manualmente suscrita en papel y en la cual se destaque el nombre del responsable, el cargo que ocupa y los datos de ubicación de la Entidad donde proviene. Pueden ser generados por una Entidad externa o dentro de la misma organización.
Estructura Departamental	Es el organigrama de tipo funcional que maneja una Entidad pública o privada.
Puesto de Trabajo	Unidad administrativa asignada a un funcionario
Funcionario	Persona que trabaja en una Entidad y que cumple un rol específico en el análisis de un trámite y tiene potestad de resolver sobre éste de acuerdo a su función.
Documento Anexo	Archivo digital que acompaña a un trámite y que surge como fruto del análisis realizado por el

	funcionario competente.
Administrador	Usuario con privilegios de establecer las configuraciones necesarias en el sistema y que le permitan registrar un trámite y canalizarlo a un funcionario para su resolución.
Cliente	Entidad que plantea un requerimiento a ser atendido por el organismo estatal. Es el gestor del trámite.
Mantenimiento	Hace referencia a las operaciones de ingreso, modificación y eliminación de datos.

Abreviaturas y Acrónimos:

ERS	Especificación de Requisitos de Software
SIGTE	Sistema de Gestión de Trámites del Estado
UML	Unified Modeling Lenguaje
RAD	Desarrollo Rápido de Aplicaciones

II.1.4 Referencias.

IEEE Recommended Practices for Requirements Specification ANSI/IEEE st. 830, 1998.

II.1.5 Visión General del documento

Este documento consta de tres secciones, la primera contiene una visión general del sistema a desarrollar. En la segunda sección se describe el sistema, sus principales funciones, gestión de los datos asociados y factores que inciden en el sistema a nivel general. Y, en la última sección se definen detalladamente los requisitos que debe satisfacer el sistema.

II.2. Descripción global

En este apartado se describe de manera general las principales funciones y restricciones que debe soportar el sistema, así como cualquier otro factor que incida en la construcción del mismo.

II.2.1 Perspectiva del producto

En una primera versión, el sistema será autónomo y no interactuará con otro aunque posteriormente deberá integrarse con un sistema de planificación, seguimiento y evaluación de proyectos, para lo cual se deberá contemplar dicha integración basado en un código de identificación del proyecto al que corresponde el trámite en caso de que el mismo tenga una resolución favorable. Caso contrario se archivará y se mantendrá como base para futuros análisis estadísticos de gestión.

II.2.2 Funciones del sistema

Lo que el sistema pretende es convertirse en una herramienta de apoyo efectivo en el proceso de registro y control de trámites, para lo cual el sistema deberá cumplir las siguientes características funcionales clasificadas por módulos:

Identificación en el sistema

Los usuarios que requieran hacer consultas sobre el estado de un trámite, podrán hacerlo de manera directa ingresando el número de trámite que será generado al momento de realizar el registro del mismo por el administrador del sistema, sin embargo el administrador y los funcionarios deberán identificarse y el sistema habilitará las funciones respectivas de acuerdo a su rol.

Catálogos

El sistema mantendrá catálogos de información y de configuración tales como: tipos de cargos, estados de trámites, prioridades, tipos de resoluciones, tipos de documentos, tipos

de entidades y tipos de trámites y que podrán ser utilizados como criterios de consulta y en la obtención de reportes.

Administración

El usuario definido como administrador, será el encargado de registrar información de la estructura departamental de la entidad, así como crear puestos de trabajo y asignarlos a los funcionarios respectivos quienes posteriormente serán los encargados de la revisión de trámites de acuerdo a su rol dentro de la organización.

Adicionalmente el sistema deberá ser georeferenciable, esto es, que permita obtener informes de trámites por provincia, cantón, parroquia, y sector o barrio de donde procede el mismo.

Trámites

El registro de trámites constituye el módulo central del sistema a desarrollarse. Esta actividad será realizada por el administrador quien deberá registrar los datos de la entidad que envía el trámite, su representante legal, el cargo y función que desempeña, el puesto y funcionario responsable a quien va dirigido el pedido, un texto con las instrucciones y pedido y la prioridad de atención requerida.

Funcionarios

Un funcionario podrá realizar el seguimiento a los trámites que le hayan sido asignados, resolver sobre la acción a tomar, añadir los anexos digitales que estime necesarios, redirigir el trámite hacia otra dependencia si fuera el caso o darlo por concluido para su archivo respectivo.

Clientes

Es un tipo de usuario que podrá consultar el estado de su trámite y no necesita estar registrado dentro del sistema para realizar esta actividad. Para ello será necesario el ingreso del número de identificación único del trámite y que será generado por el sistema al momento del ingreso.

Consultas

Deberá existir un medio de consultas de los trámites bajo diferentes criterios y cuya información podrá ser exportada a un generador de cuadros estadísticos u hoja electrónica.

II.2.3 Características del usuario

Se prevé que los usuarios del sistema sean de diversa formación desde aquellos que conocen el uso de las tecnologías de la información hasta aquellos que recién se encuentren conociendo sus características y beneficios.

El sistema manejará los siguientes tipos de usuarios:

Administrador: Será el encargado de registrar los datos que sirven como parámetros del sistema, definirá departamentos, establecerá los datos de georeferenciación (provincias, cantones, parroquias, sector/barrio), registrará entidades, funciones, puestos de trabajo, y los datos de ingreso y cambio de estado de los trámites.

Funcionarios: serán las personas quienes realizarán las actividades de seguimiento y resolución de trámites. Tendrán además la facultad de anexar documentos en formato digital que consideren necesarios en cada actividad.

Clientes: consultarán el estado de un trámite y su resolución. Este tipo de usuario puede no tener mayor formación en el manejo de sistemas por lo que esta actividad deberá realizarla a través de una interfaz intuitiva y de fácil manejo.

II.2.3 Restricciones generales.

El presente proyecto no tiene restricciones, excepto por el tiempo de desarrollo, para lo cual se plantea el uso de una herramienta RAD que genere la mayor cantidad de código fuente donde sea factible.

II.2.5 Suposiciones y dependencias

II.2.5.1 Suposiciones

Los requisitos establecidos en este documento serán estables una vez aprobados por el cliente, sin embargo cualquier cambio que se introduzca posteriormente, deberá ser aprobado por las partes implicadas en el desarrollo y actualizados por el grupo de gestión de la configuración.

II.2.5.2 Dependencias

La primera versión del sistema deberá permitir visualizar los documentos anexos a un trámite mediante el uso de programas de ofimática comunes como Microsoft Office o Acrobat. Así mismo se considerará un medio de enlace común hacia un sistema de seguimiento y evaluación de proyectos a implementarse posteriormente, pero que no será desarrollado en esta etapa.

II.3 Requisitos específicos

En este apartado se indican a detalle los requisitos que deberá satisfacer el sistema, y que son esenciales para el desarrollo.

II.3.1 Requisitos de las interfaces externas

II.3.1.1 Interfaces de usuario

El Internet es el medio más efectivo de masificación de los servicios, por lo tanto el sistema estará desarrollado sobre plataforma web, y se mostrará al usuario mediante el uso de un browser que soporte la arquitectura que se defina.

II.3.1.3 Interfaces de hardware

No se ha detallado.

II.3.1.4 Interfaces de software

Se implementará sobre Internet como medio de acceso, por lo que se considera el uso de servidores y navegadores web.

II.3.1.5 Interfaces de comunicaciones

El sistema será accedido a través de computadores conectados a Internet sobre el protocolo de comunicaciones TCP/IP (Transmisión Control Protocol/Internet Protocol) de forma remota, o mediante conexión directa a la red Ethernet disponible en la entidad.

II.3.2 Requisitos funcionales

Identificación en el sistema

Requisito 1: El sistema solicitará una clave de acceso, la misma que será única para cada usuario y permitirá habilitar las opciones del sistema de acuerdo al rol del mismo dentro de la entidad. En caso de no ingresar una clave, se mostrará una interfaz de consulta de trámites.

Gestión Catálogos

Requisito 2: Se permitirá mantener (adicionar, modificar, eliminar) un catálogo de cargos que podrán ser asignados a un funcionario o a un representante legal de una entidad que suscriba un trámite.

Requisito 3: El sistema permitirá realizar el mantenimiento de la descripción de los estados de un trámite de acuerdo a lo que se establece en el Req. 17, 18, 19.

Requisito 4: Se podrán definir y mantener las prioridades de atención de los trámites. Estos pueden ser: alta, normal o baja.

Requisito 5: El sistema permitirá mantener un listado de resoluciones para un trámite. Por ejemplo: tramitar, preparar informe, archivar, pagar.

Requisito 6: Se permitirá definir tipos de documentos que serán anexados a los trámites. Por ejemplo: contrato, informe técnico, términos de referencia.

Requisito 7: Se permitirá establecer diferentes tipos de Entidades. Por ejemplo: gremio de profesionales, gremio de agricultores, universidad, junta parroquial.

Requisito 8: Se podrá mantener tipos de trámites. Por ejemplo: interno, externo

Administración

Requisito 9: Se permitirá definir la estructura departamental de la entidad, la misma que deberá ser definida con un orden de jerarquía de n niveles, y se identificará al nivel como un campo numérico, donde el nivel 0, corresponderá al primer elemento de la jerarquía. 1, si es un departamento que tiene subdepartamentos de nivel inferior, y 2 si es el último nivel de la jerarquía, es decir que no tiene subdepartamentos de nivel inferior. Se registrarán la descripción del departamento, el nivel, y el subdepartamento al que pertenece. Podrán asignarse puestos de trabajo únicamente a los departamentos cuyo nivel sea 2 de acuerdo al Req 10.

Requisito 10: Se establecerán puestos de trabajo para cada departamento de nivel inferior en concordancia con Req9. Donde se especificará la denominación del puesto y opcionalmente una dirección de correo electrónico.

Requisito 11: El sistema permitirá establecer las funciones correspondientes a un puesto de trabajo para el personal registrado. Se seleccionará el puesto de trabajo definido en

Req10 y se listará los funcionarios en base a lo que establece Req25. Opcionalmente se podrá registrar una observación del motivo del alta.

Requisito 12: Una vez registrado el nuevo funcionario, el sistema automáticamente registrará la fecha y hora de ingreso y lo marcará como Activo dentro del sistema. Se podrá suspender a un funcionario de un cargo, el sistema deberá registrar automáticamente la fecha de salida del mismo. En el caso de que un funcionario se encuentre previamente asignado a un puesto de trabajo y se le asigne otra función, el sistema automáticamente deberá suspender al funcionario del puesto anterior y activarlo en el nuevo puesto registrándose además de forma automática, la fecha y hora de salida e ingreso respectivos.

Requisito 13: Únicamente aquellos funcionarios que se encuentren asignados a un puesto de trabajo, podrán recibir un trámite y aparecerán en el campo de responsables del trámite en concordancia con Req. 26.

Requisito 14: El sistema permitirá mantener una jerarquía de referencias geográficas en base al orden provincia, cantón y parroquia. Es decir, en base a la información de una parroquia se podrá establecer el cantón a la que pertenece y su provincia o viceversa. Adicionalmente se permitirá mantener un listado de sectores o barrios independiente.

Gestión de Trámites

Requisito 15: El sistema mantendrá un registro de Entidades donde se especificará el nombre de la entidad, el tipo de entidad de acuerdo al catálogo establecido en Req 7, la parroquia y la localidad a la que pertenece en base a lo que establece Req 14, opcionalmente se podrá registrar los datos de contacto: dirección, teléfono, fax y correo electrónico.

Requisito 16: Se permitirá el registro de trámites, donde se deberá seleccionar la fecha de ingreso del documento desde un calendario que por defecto mostrará la fecha actual, el

tipo de trámite en base al catalogo establecido en Req 8, se deberá seleccionar la entidad que envía el trámite (Req. 15), el representante legal de acuerdo al listado de personas registrado en Req 27, el cargo que desempeña (Req. 2), Un texto libre y opcional para identificar la función que desempeña el representante legal al momento del registro, el puesto y el funcionario responsable de ese puesto y en concordancia a los Req. 10 y Req 11. y Req 25 , un texto libre para el registro de las instrucciones pertinentes, la prioridad de atención del trámite en base a Req 4. Un campo de texto obligatorio donde se deberá detallar el motivo o asunto del trámite, adicionalmente un campo opcional de texto donde se indicará el archivo físico del trámite y finalmente un campo de texto libre donde se podrá indicar observaciones adicionales.

Requisito 17: La primera vez que se registre un trámite, automáticamente el sistema deberá grabarlo con estado de *Registro*. En este estado, podrá ser factible modificar la información del tramite (Req. 16).

Requisito 18: El administrador podrá pasar el trámite del estado de *Registro* al estado de *Revisión.* En este estado no será factible realizar ningún cambio en el trámite, y adicionalmente el trámite podrá ser visualizado como en *Revisión* de acuerdo a Req 21, y el funcionario responsable a quien le corresponda resolver las acciones a tomar, podrá despacharlo de acuerdo a lo que se indica en Req 22.

Requisito 19: El administrador podrá pasar el trámite del estado de *Revisión* al estado *Concluido* siempre y cuando no exista ninguna acción pendiente por resolver por un funcionario (Req. 21, Req 22) y que deberá ser controlado por el sistema. Registrará opcionalmente el archivo físico y las observaciones que estime pertinentes. El sistema registrará automáticamente la fecha de cierre y a partir de ese momento, todos los datos del trámite serán de consulta y no podrán ser modificados.

Requisito 20: El sistema generará automáticamente un número de identificación único para cada trámite al momento de su registro y adicionalmente almacenará

automáticamente la fecha y hora de registro del trámite, la misma que puede diferir de la fecha de ingreso del documento al sistema.

Requisito 21: Un funcionario podrá visualizar un listado y la cantidad de trámites en estado de revisión que le hayan sido asignados al momento de registro del trámite (Req. 16). Podrá enviar nuevas instrucciones a otros puestos de trabajo y funcionarios responsables para lo cual el sistema presentará el listado de puestos y responsables de acuerdo a Req 11, un campo de texto donde podrá ingresar las instrucciones pertinentes y finalmente el tipo de resolución en base al catálogo establecido en Req.5

Requisito 22: Si un trámite ha sido despachado, el funcionario responsable podrá identificarlo, en este caso el trámite desaparecerá del listado de trámites en estado de revisión (Req. 21) y si no existen más acciones pendientes sobre el trámite que hayan sido asignados a otros funcionarios, el administrador podrá cerrarlo como se indica en Req 19.

Requisito 23. El funcionario responsable de resolver sobre un trámite, podrá anexar la documentación que estime pertinente en formato de texto digital. Para lo cual se registrará el nombre del documento, la ubicación física en el servidor, el tipo de documento en base al catálogo definido en Req 6, y un comentario opcional. Estos documentos podrán ser visualizados en el mismo sistema.

Requisito 24. Se podrán registrar varios documentos anexos por cada acción pendiente para un trámite (Req. 23). Cuando la acción sea cambiada a estado despachado, ya no será factible adicionar nuevos documentos (Req 22).

Gestión de Funcionarios

Requisito 25. Se registrarán funcionarios donde se especificarán sus datos personales: nombres, apellidos, código de identificación único para acceso al sistema, domicilio, teléfono de contacto, número de teléfono móvil, correo electrónico.

Requisito 26. Los funcionarios podrán ingresar con su código de identificación establecido en Req. 25 y podrán gestionar y resolver los trámites que les hayan sido asignados, así como anexar documentos, en concordancia con Req 21, Req 22 y Req 23.

Clientes

Requisito 27: Se podrán registrar personas que suscriben los trámites y se almacenarán sus datos personales que serán los mismos establecidos en Req 25, excepto por el código de identificación que no deberá ser registrado.

Consultas

Requisito 28: El administrador y los funcionarios podrán realizar consultas de documentos utilizando como filtro cualquiera de los parámetros establecidos en los Req 1 al Req 8, Req 14. Además del número de trámite, fecha de registro y asunto.

Requisito 29: Los usuarios del sistema podrán consultar los datos de registro y resolución de trámites ingresando únicamente su código de identificación, de acuerdo a lo que establece Req 20.

Requisito 30: El administrador y los funcionarios podrán visualizar los documentos de texto anexos en formato digital, así como podrán exportar dicha información hacia una hoja electrónica para el análisis respectivo. Un usuario no podrá ver esta información.

II.3.3 Requisitos de rendimiento

Al ser un sistema que puede ser consultado desde una red local o de área extendida, el rendimiento mucho dependerá de las velocidades de transmisión y del hardware de servidores sobre los cuales se soporte al sistema, los mismos que no están definidos en

esta fase. Sin embargo se esperan tiempos de respuesta que no superen los 10 segundos sobre redes extendidas y menores que 3 segundos sobre redes locales.

II.3.4 Requisitos tecnológicos

Los requisitos tecnológicos no se especifican para la plataforma de soporte, sin embargo se estima que los usuarios dispongan de terminales con la configuración mínima requerida por los navegadores web.

II.3.6 Requisitos de seguridad

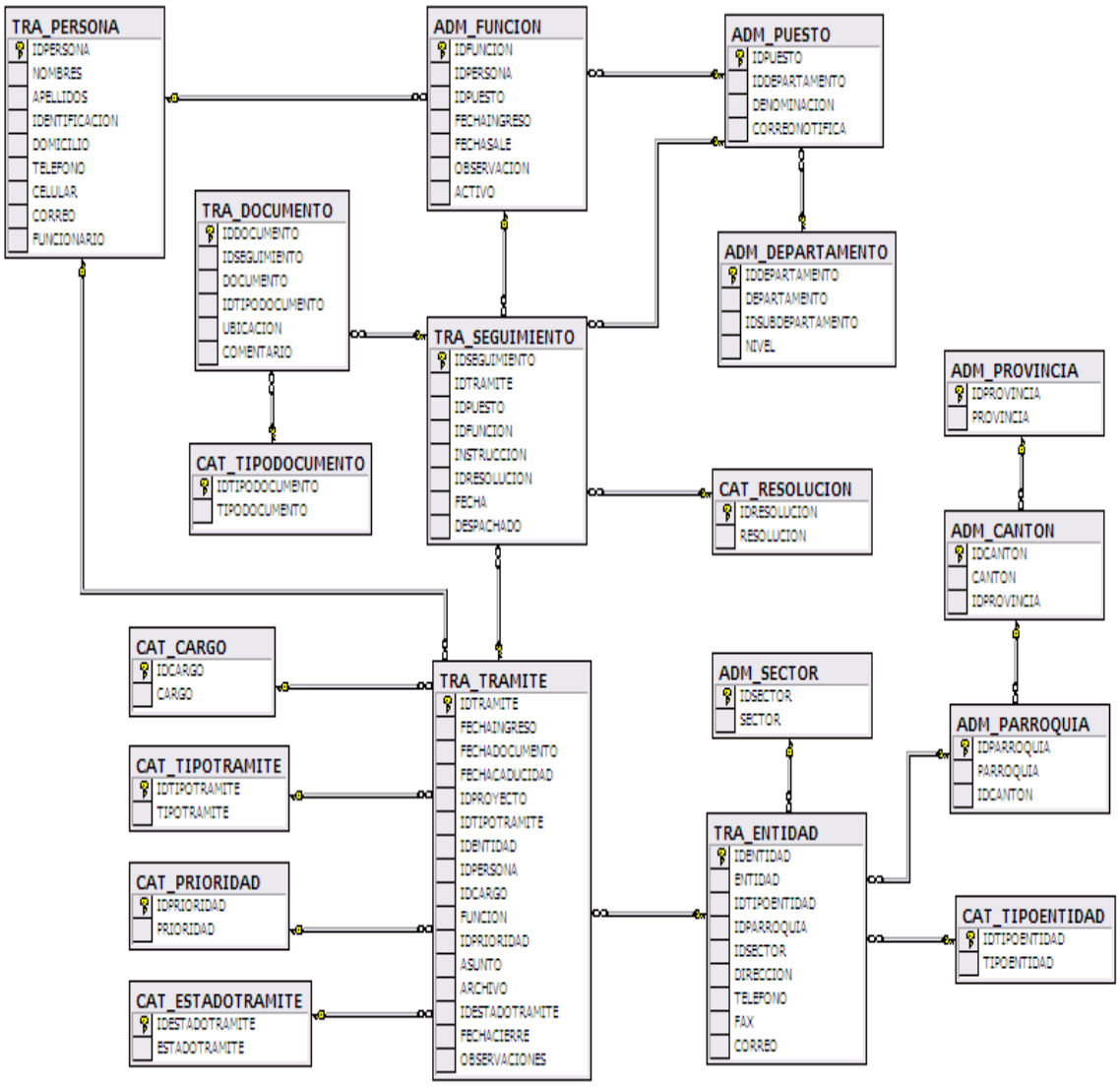
Se utilizarán técnicas que permitan la construcción de sitios web seguros que garanticen la privacidad de la información donde sea necesario.

La integridad y seguridad de los datos corresponden al administrador del sistema y se generarán pistas de auditoria para las transacciones críticas como son el registro de trámites y resolución por cada funcionario.

ANEXO C
MÉTRICAS DE ESTIIMACIÓN

MÉTRICA DE PUNTOS DE FUNCIÓN CONVENCIONAL

MODELO ENTIDAD RELACION SISTEMA SIGTE



ANÁLISIS DE LOS ELEMENTOS DE FUNCIÓN DEL PROYECTO SIGTE

Elemento de Función	Funciones	Tablas Afectadas FTRs	Tipos de Elementos de Datos DETs	Complejidad ILFs
Consulta Externa	Solicitar_Clave	1	2	Baja
Entrada Externa	Adicionar_Persona	1	10	Baja
Entrada Externa	Modificar_Persona	1	10	Baja
Entrada Externa	Eliminar_Persona	1	10	Baja
Consulta Externa	Consultar_Persona	1	10	Baja
Entrada Externa	Adicionar_Cargo	1	3	Baja
Entrada Externa	Modificar_Cargo	1	3	Baja
Entrada Externa	Eliminar_Cargo	1	3	Baja
Consulta Externa	Consultar_Cargo	1	3	Baja
Entrada Externa	Adicionar_EstadoTramite	1	3	Baja
Entrada Externa	Modificar_EstadoTramite	1	3	Baja
Entrada Externa	Eliminar_EstadoTramite	1	3	Baja
Consulta Externa	Consultar_EstadoTramite	1	3	Baja
Entrada Externa	Adicionar_Prioridad	1	3	Baja
Entrada Externa	Modificar_Prioridad	1	3	Baja
Entrada Externa	Eliminar_Prioridad	1	3	Baja
Consulta Externa	Consultar_Prioridad	1	3	Baja
Entrada Externa	Adicionar_Resolucion	1	3	Baja
Entrada Externa	Modificar_Resolucion	1	3	Baja

Entrada Externa	Eliminar_Resolucion	1	3	Baja
Consulta Externa	Consultar_Resolucion	1	3	Baja
Entrada Externa	Adicionar_Documento	3	7	Alta
Entrada Externa	Modificar_Documento	3	7	Alta
Consulta Externa	Consultar_Documento	3	7	Media
Entrada Externa	Adicionar_TipoDocumento	1	3	Baja
Entrada Externa	Modificar_TipoDocumento	1	3	Baja
Consulta Externa	Consultar_TipoDocumento	1	3	Baja
Entrada Externa	Adicionar_TipoEntidad	1	3	Baja
Entrada Externa	Modificar_TipoEntidad	1	3	Baja
Entrada Externa	Eliminar_TipoEntidad	1	3	Baja
Consulta Externa	Consultar_TipoEntidad	1	3	Baja
Entrada Externa	Adicionar_TipoTramite	1	3	Baja
Entrada Externa	Modificar_TipoTramite	1	3	Baja
Entrada Externa	Eliminar_TipoTramite	1	3	Baja
Consulta Externa	Consultar_TipoTramite	1	3	Baja
Entrada Externa	Adicionar_Departamento	1	5	Baja
Entrada Externa	Modificar_Departamento	1	5	Baja
Entrada Externa	Eliminar_Departamento	1	5	Baja
Consulta Externa	Consultar_Departamento	1	5	Baja
Salida Externa	Desplegar_EstructuraDepartamental	1	4	Baja
Entrada Externa	Adicionar_PuestoTrabajo	2	5	Baja

Entrada Externa	Modificar_PuestoTrabajo	2	5	Baja
Entrada Externa	Eliminar_PuestoTrabajo	2	5	Baja
Entrada Externa	Adicionar_Funcion	3	8	Alta
Entrada Externa	Modificar_Funcion	3	8	Alta
Entrada Externa	Eliminar_Funcion	3	8	Alta
Consulta Externa	Listar_FuncionActivo	3	8	Media
Entrada Externa	Registrar_Tramite	7	17	Alta
Entrada Externa	Modificar_Tramite	7	17	Alta
Consulta Externa	Consultar_TramiteRevision	7	17	Alta
Salida Externa	Mostrar_NumeroTramitesPendientes	1	1	Baja
Entrada Externa	Ingresar_SeguimientoTramite	5	9	Alta
Entrada Externa	Modificar_SeguimientoTramite	5	9	Alta
Entrada Externa	Eliminar_SeguimientoTramite	5	9	Alta
Consulta Externa	Consultar_SeguimientoTramite	5	9	Alta
Entrada Externa	Adicionar_Parroquia	2	4	Baja
Entrada Externa	Modificar_Parroquia	2	4	Baja
Entrada Externa	Eliminar_Parroquia	2	4	Baja
Consulta Externa	Consultar_Parroquia	2	4	Baja
Entrada Externa	Adicionar_Canton	2	4	Baja
Entrada Externa	Modificar_Canton	2	4	Baja
Entrada Externa	Eliminar_Canton	2	4	Baja
Consulta Externa	Consultar_Canton	2	4	Baja

Entrada Externa	Adicionar_Provincia	1	3	Baja
Entrada Externa	Modificar_Provincia	1	3	Baja
Entrada Externa	Eliminar_Provincia	1	3	Baja
Consulta Externa	Consultar_Provincia	1	3	Baja
Entrada Externa	Adicionar_Sector	1	3	Baja
Entrada Externa	Modificar_Sector	1	3	Baja
Entrada Externa	Eliminar_Sector	1	3	Baja
Consulta Externa	Listar_Sectores	1	3	Baja
Entrada Externa	Adicionar_Entidad	4	10	Alta
Entrada Externa	Modificar_Entidad	4	10	Alta
Entrada Externa	Eliminar_Entidad	4	10	Alta
Consulta Externa	Consultar_Entidad	4	10	Alta
Consulta Externa	Listar_DocumentoAnexos	3	7	Media
Consulta Externa	Visualizar_DocumentoAnexo	1	2	Baja
Interfaz Externa	Calendario	1	3	Baja

Archivos Lógicos ILFs	**Registros Lógicos**	**Tipos de Elementos de Datos DETs**	**Complejidad ILFs**
TRA_PERSONA	1	9	Baja
TRA_SEGUIMIENTO	1	8	Baja
TRA_TRAMITE	1	16	Baja
TRA_ENTIDAD	1	9	Baja
ADM_FUNCION	1	7	Baja

ADM_PUESTO	1	4	Baja
ADM_DEPARTAMENTO	1	4	Baja
ADM_PROVINCIA	1	2	Baja
ADM_CANTON	1	3	Baja
ADM_PARROQUIA	1	3	Baja
ADM_SECTOR	1	2	Baja
CAT_RESOLUCION	1	2	Baja
CAT_TIPOENTIDAD	1	2	Baja
CAT_CARGO	1	2	Baja
CAT_TIPOTRAMITE	1	2	Baja
CAT_PRIORIDAD	1	2	Baja
CAT_ESTADOTRAMITE	1	2	Baja
TRA_DOCUMENTO	1	6	Baja
CAT_TIPODOCUMENTO	1	2	Baja

ENTRADAS EXTERNAS

Archivos referenciados	**Elementos de Datos**		
	1-4	5-15	>15
0-1	Baja(29)	Baja(3)	Media
2	Baja(9)	Media	Alta
3 ó más	Media	Alta(11)	Alta(2)

SALIDAS EXTERNAS

Archivos referenciados	**Elementos de Datos**		
	1-5	6-19	>19
0-1	Baja(2)	Baja	Media
2-3	Baja	Media	Alta
>3	Media	Alta	Alta

CONSULTAS EXTERNAS

Archivos referenciados	Elementos de Datos		
	1-5	6-19	>19
0-1	Baja(12)	Baja(1)	Media
2-3	Baja(2)	Media(3)	Alta
>3	Media	Alta(3)	Alta

ARCHIVOS LÓGICOS INTERNOS

Tipos de Registros	Elementos de Datos		
	1-19	20-50	>50
1	Baja(19)	Baja	Media
2-5	Baja	Media	Alta
>5	Media	Alta	Alta

INTERFASES EXTERNAS

Tipos de Registros	Elementos de Datos		
	1-19	20-50	>50
1	Baja(1)	Baja	Media
2-5	Baja	Media	Alta
>5	Media	Alta	Alta

PUNTOS DE FUNCION

Punto de Función	Complejidad			
	Alta	**Media**	**Baja**	**Total**
Entradas Externas	6***13**	4	3***41**	201
Salidas Externas	7	5	4***2**	8
Consultas Externas	6***3**	4***3**	3***15**	75
Archivos Lógicos Internos	15	10	7***19**	133
Interfases Externas	10	7	5***1**	5
			PFSA	**422**

CARACTERISTICAS GENERALES DEL SISTEMA

Valor	Significado
0	Sin influencia, factor no presente
1	Influencia insignificante, muy baja
2	Influencia moderada o baja
3	Influencia media, normal
4	Influencia alta, significativa
5	Influencia muy alta, esencial

	Características Generales del Sistema	Complejidad	Comentario
1	Comunicación de datos	3	Comunicación de datos moderada.
2	Procesamiento de datos distribuido	1	Sistema distribuido con conexión remota o directa a la red.
3	Rendimiento	2	Tiempos de respuesta que no superen los 10 segundos en redes extendidas y menores que 3 segundos sobre redes locales.
4	Uso de hardware existente	1	No se ha detallado.
5	Transacciones	3	Moderado manejo de transacciones.
6	Entrada de datos interactiva	4	Manejo de browser de Internet para la entrada de datos.
7	Eficiencia	4	Si existe eficiencia del sistema.
8	Actualizaciones on-line	4	Manejo de Internet para ejecutar transacciones.
9	Complejidad de procesamiento	1	No hay cálculos en el sistema.
10	Reusabilidad	0	No se requiere que el código sea reutilizable
11	Facilidad de conversión e instalación	3	Los usuarios disponen de terminales con la configuración mínima requerida por los navegadores web.
12	Facilidad de operación	3	Operación normal.
13	Múltiples instalaciones	1	Se requiere escasas instalaciones del sistema en todos los equipos.
14	Facilidad de mantenimiento	3	Se requiere un costo moderado de mantenimiento.
Valor de Complejidad VAF		**33**	

CÁLCULOS DE PUNTOS DE FUNCION AJUSTADO:

PFSA= Punto de Función sin ajustar= 422

PFA= PFSA *((VAF*0.01)+0.65)

VAF= Grado Total de Complejidad= 33

- PFA= 422((33*0.01)+0.65)= 413.56

TRANSFORMACIÓN PF A SLOC

Size = 32 (VB) x 422 PFSA = 13504 SLOC = **13.504 KSLOC**

CÁLCULO DE ESFUERZO NOMINAL

$$PM_{nominal} = A \times (Size)^{B}$$

AJUSTE DE FACTORES DE ESCALA (B)

FÓRMULA:

$$B = 1.01 + 0.01 \times \sum_{j=1}^{5} Wj$$

VARIABLE	DESCRIPCION	PODERACIÓN	VALOR
PREC	El sistema es familiar.	Alto	2.48
FLEX	La flexibilidad de desarrollo es moderada.	Nominal	3.04
RESL	La arquitectura es sólida y los riesgos generalmente se mitigan.	Alto	2.83
TEAM	La interacción del equipo es cooperativa.	Alto	2.19
PMAT	La madurez del proceso software es normal.	Nominal	4.68
		TOTAL	**1.16**

$PM_{nominal}$= 2.94 x $(13.504)^{1.16}$

$PM_{nominal}$= 60.21 Meses-Persona

CÀLCULO DE ESFUERZO AJUSTADO

$$PM_{ajustado} = PM_{nominal} \times \Pi(ME_i)$$

AJUSTE DE DRIVERS DE COSTO

FÓRMULA:

$$\Pi(ME_i)$$

MULTIPLICADOR	DESCRIPCION	PONDERACIÓN	VALOR
PERS	Se tiene analistas y programadores con gran eficiencia y buen trabajo en equipo. Dedicación full-time.	Alto	0.83
RCPX	La exigencia de confiabilidad, documentación y volumen de datos son moderadas, el producto no tiene mucha complejidad.	Nominal	1
RUSE	No se pretende reutilizar nada.	Bajo	0.95
PDIF	Existen restricciones al tiempo de respuesta. No se especifica el hardware a utilizar, ni la plataforma de soporte.	Bajo	0.87
PREX	Todos los analistas y programadores tienen una moderada experiencia de desarrollo de la aplicación.	Bajo	1.12
SCED	Se requiere terminar el proyecto en el tiempo estimado.	Nominal	1
FCIL	Se requiere herramienta CASE para desarrollo Web. La comunicación de datos debe ser distribuido con conexión a Internet.	Alto	0.87
		TOTAL	**0.67**

$PM_{ajustado}$= 60.21 x 0.67

$PM_{ajustado}$= 40.34 Meses-persona

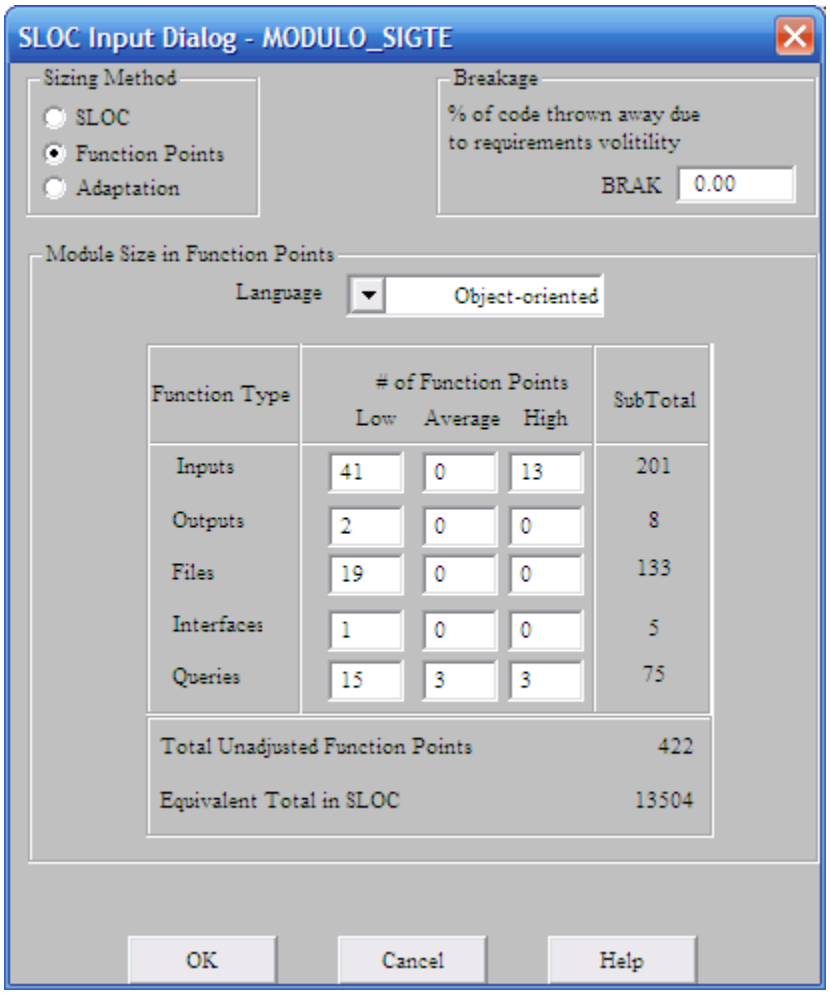
CÁLCULO DEL TIEMPO DE DESARROLLO

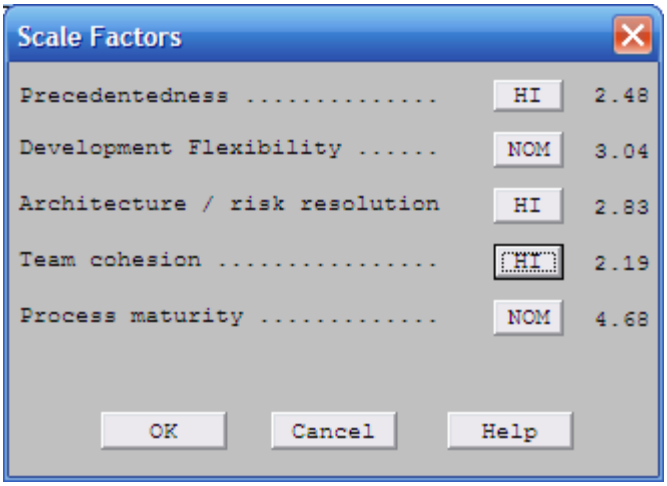
$$TDEV = [3.67 \times PM^{(0.28+0.2\times(B-1.01))}] \times \frac{SCED\%}{100}$$

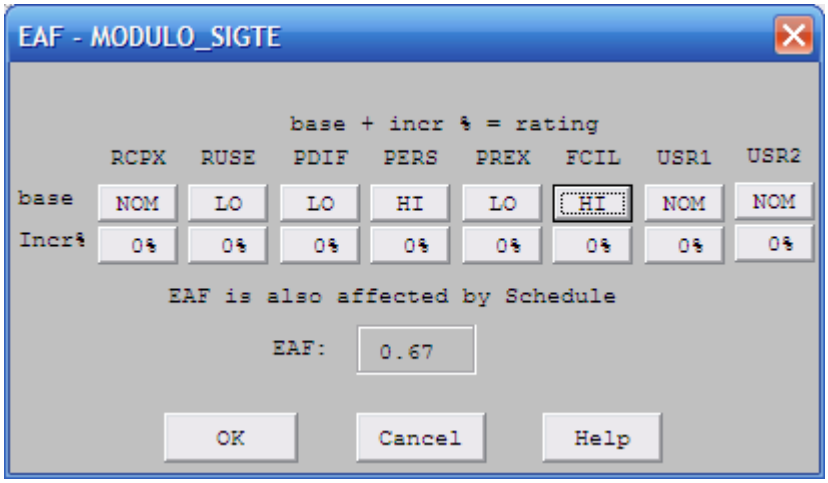
TDEV= [3.67 X 40.34 $^{(0.28+0.2x(1.16-1.01))}$] x 1

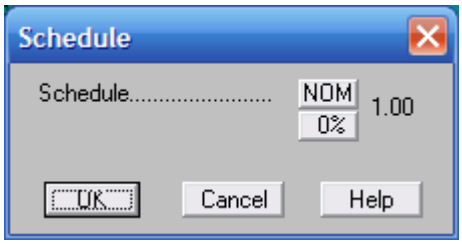
TDEV= [3.67 X 40.34 $^{0.31}$] x 1 = **11.55 Meses**

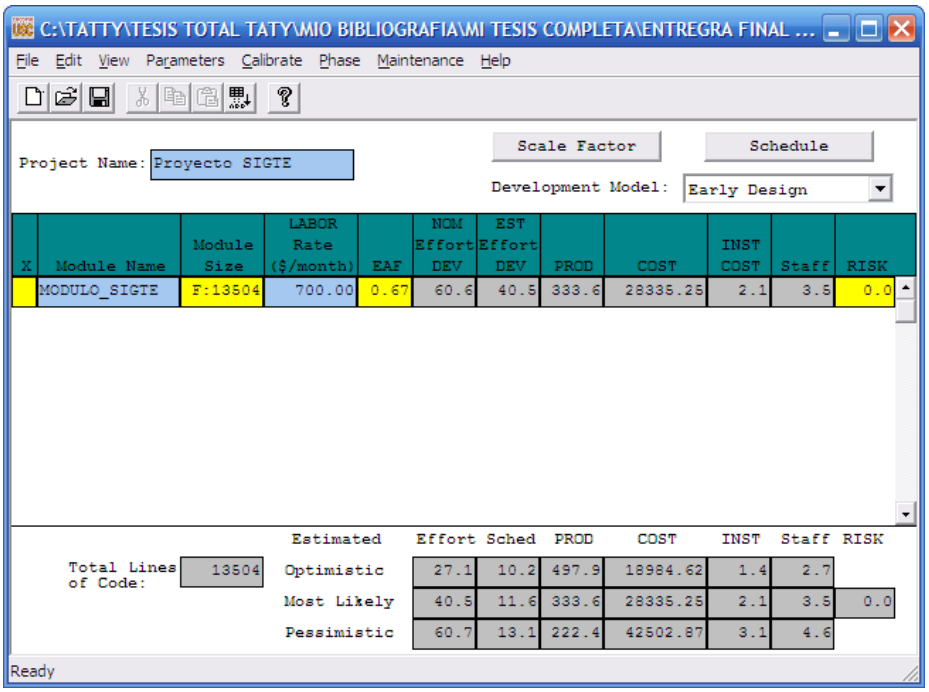
CÁLCULO MEDIANTE HERRAMIENTA COCOMO





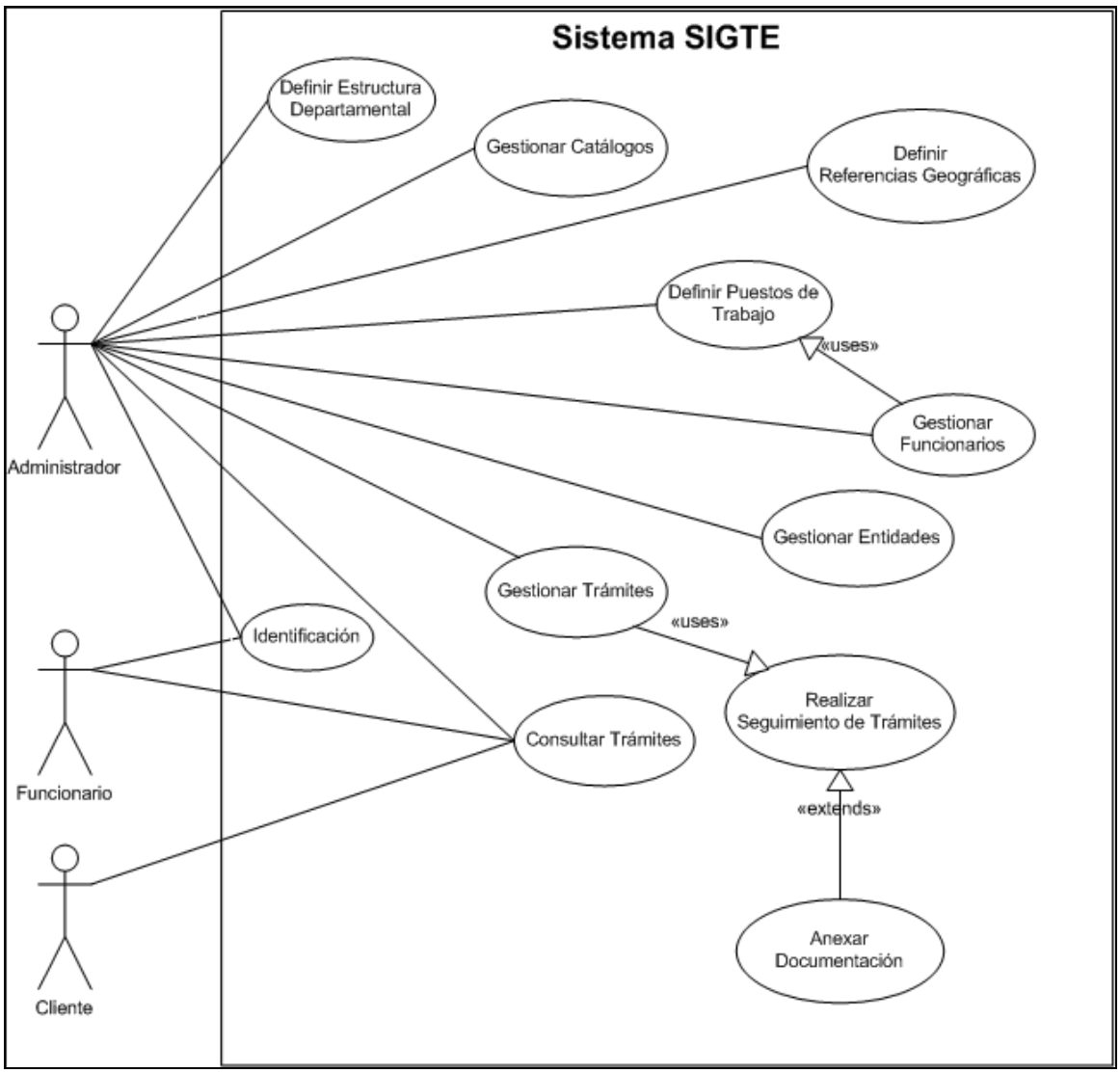

EAF - MODULO_SIGTE
base + incr % = rating
RCPX RUSE PDIF PERS PREX FCIL USR1 USR2
base NOM LO LO HI LO HI NOM NOM
Incr% 0% 0% 0% 0% 0% 0% 0% 0%
EAF is also affected by Schedule
EAF: 0.67
OK
Cancel
Help


Schedule
Schedule..... NOM 0% 1.00
OK
Cancel
Help


C:\TATTY\TESIS TOTAL TATY\MIO BIBLIOGRAFIA\MI TESIS COMPLETA\ENTREGRA FINAL ...
File Edit View Parameters Calibrate Phase Maintenance Help
Project Name: Proyecto SIGTE
Scale Factor
Schedule
Development Model: Early Design
X Module Name Module Size LABOR Rate ($/month) EAF NOM Effort DEV EST Effort DEV PROD COST INST COST Staff RISK
MODULO_SIGTE F:13504 700.00 0.67 60.6 40.5 333.6 28335.25 2.1 3.5 0.0
Estimated Effort Sched PROD COST INST Staff RISK
Total Lines of Code: 13504
Optimistic 27.1 10.2 497.9 18984.62 1.4 2.7
Most Likely 40.5 11.6 333.6 28335.25 2.1 3.5 0.0
Pessimistic 60.7 13.1 222.4 42502.87 3.1 4.6
Ready

TÉCNICA PUNTOS DE CASOS DE USO

DIAGRAMA DE CASOS DE USO SISTEMA SIGTE



Total Número de Actores: 3

Total Número de Casos de Uso: 11

Factor de Peso de los Actores sin ajustar (UAW)

Tipo de Actor	Descripción	Factor de Peso
Simple	Otro sistema que interactúa con el sistema a des arrollar mediante una interfaz de programación (API, Application Programming Interface)	1
Medio	Otro sistema que interactúa con el sistema a des-arrollar mediante un protocolo o una interfaz basada en texto	2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica	3

- UAW= 3x3= 9

Factor de Peso de los Casos de Uso sin ajustar (UUCW)

Los casos de uso "Definir Estructura Departamental", "Identificación", "Consultar Trámites" y "Definir Referencias Geográficas" consisten en una única transacción.

Los siguientes casos de uso "Gestión Catálogos" consiste en tres transacciones de gestión (adicionar, modificar, eliminar), "Gestionar Entidades" consiste en cuatro transacciones de gestión (adicionar, modificar, eliminar, consultar). Mientras que el caso de uso "Definir Puesto de Trabajo" consiste en una sola transacción pero requiere realizar la transacción "Gestionar Funcionarios" (transacciones: adicionar, modificar, eliminar, consultar); la transacción "Gestionar Trámites" (transacciones: adicionar, modificar, eliminar, consultar) requiere realizar la transacción "Realizar Seguimiento de Trámites" (transacciones: adicionar, modificar, eliminar, consultar), y ésta a su vez "Anexar Documentación" si es necesario (transacciones: adicionar, modificar, eliminar, consultar).

Tipo de Caso de Uso	Descripción	Factor de Peso
Simple	El Caso de Uso contiene de 1 a 3 transacciones	5
Medio	El Caso de Uso contiene de 4 a 7 transacciones	10
Complejo	El Caso de Uso contiene más de 8 transacciones	15

Tipos de Casos de Uso: 6 Simples
Factor de Peso: 5

Tipos de Casos de Uso: 5 Medio
Factor de Peso: 10

- UUCW= (6x5)+ (5x10)= 80

Cálculo de Puntos de Casos de Uso no ajustados

- UUCP = UAW + UUCW
- UUCP= 9+80

- UUCP= 89

Cálculo de Puntos de Casos de Uso ajustados

UCP = UUCP x TCF x EF

Factor de complejidad técnica (TCF)

Factor	Descripción	Peso	Valor Asignado	Comentario
T1	Sistema distribuido	2	1	Sistema distribuido con conexión remota o directa a la red.
T2	Objetivos de performance o tiempo de respuesta	1	2	Tiempos de respuesta que no superen los 10 segundos en redes extendidas y menores que 3 segundos sobre redes locales.
T3	Eficiencia del usuario final	1	4	Si existe eficiencia en el manejo del sistema
T4	Procesamiento interno complejo	1	1	Hay pocos cálculos en el sistema.
T5	El código debe ser reutilizable	1	0	No se requiere que el código sea reutilizable.
T6	Facilidad de instalación	0.5	3	Los usuarios disponen de terminales con la configuración mínima requerida por los navegadores web.
T7	Facilidad de uso	0.5	3	Normal.
T8	Portabilidad	2	0	No se especifica la plataforma de soporte.
T9	Facilidad de cambio	1	3	Se requiere un costo moderado de mantenimiento.
T10	Concurrencia	1	1	Baja concurrencia de múltiples procesos.
T11	Incluye objetivos especiales de seguridad	1	5	Sitios web seguros que garanticen la privacidad de la información.
T12	Provee acceso directo a terceras personas	1	3	Los usuarios requieren identificación.
T13	Se requieren facilidades especiales de entrenamiento a usuarios	1	3	Moderado entrenamiento para el administrador, funcionario y cliente.
	SUMATORIA = 27			

$$TCF = 0.6 + 0.01 \times \Sigma (Peso_i \times Valor\ asignado_i)$$

- TCF = 0.6 + 0.01 x 27 = 0.87

Factor de ambiente (EF)

Factor	Descripción	Peso	Valor Asignado	Comentario
E1	Familiaridad con el modelo de proyectos utilizado	1.5	4	El grupo está bastante familiarizado con el modelo
E2	Experiencia en la aplicación	0.5	3	La mayoría del grupo tiene experiencia moderada en el desarrollo de estas aplicaciones
E3	Experiencia en orientación a objetos	1	4	La mayoría del grupo utiliza programación orientada a objetos
E4	Capacidad del analista líder	0.5	4	Se tiene un analista líder con experiencia
E5	Motivación	1	5	El grupo está altamente motivado
E6	Estabilidad de los requerimientos	2	5	No se esperan cambios
E7	Personal part-time	-1	0	Todo el grupo trabaja a tiempo completo
E8	Dificultad del lenguaje de programación	-1	3	El sistema estará desarrollado sobre plataforma web

$$EF = 1.4 - 0.03 \times \Sigma (Peso_i \times Valor\ asignado_i)$$

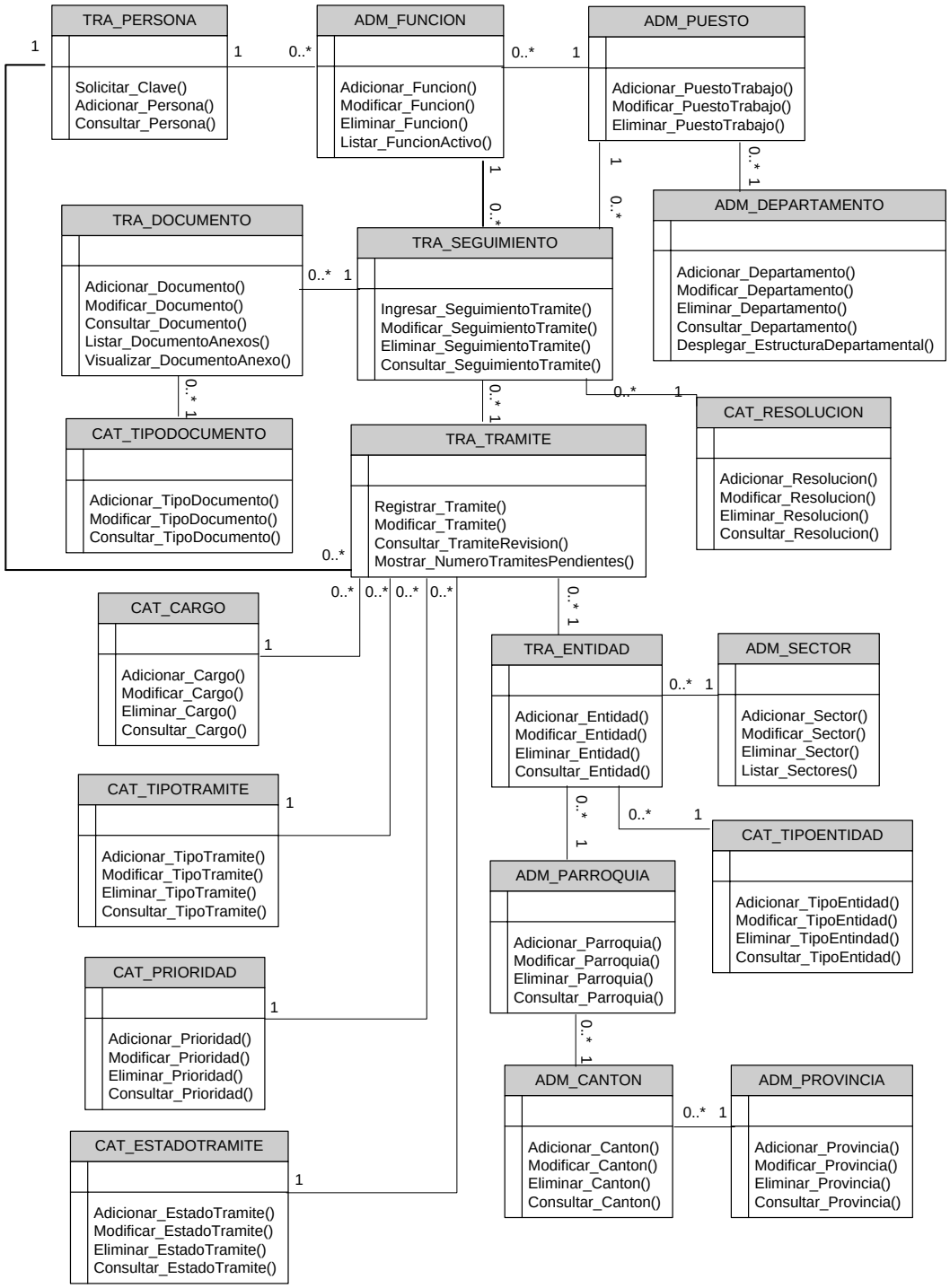
- EF = 1.4 - 0.03 x 25.5 = 0.64

PUNTOS DE CASOS DE USO AJUSTADOS

- UCP = 89 * 0.87 * 0.64 = 49.56

MEDICIÓN DE MODELOS CONCEPTUALES BASADO EN EVENTOS Y ORIENTADO A OBJETOS

MODELO ESTRUCTURAL SIGTE



- **Tamaños basados en OET**

MEASUREMENT OBJECT	MEASURE DESCRIPTION	MEASURE DEFINITION
Object type: $P \in T$	Count of Event Participations	$CEP(P) = \#\{e \in A \mid \tau(e,P) \neq \text{'/'}\}$
Conceptual model: S	Level of Object-Event Interaction	$LOEI(S) = \sum_{P \in T} CEP(P)$

TABLA OET DEL SISTEMA SIGTE

EVENTOS	TRA_PERSONA	TRA_SEGUIMIENTO	TRA_TRAMITE	TRA_ENTIDAD	ADM_FUNCION	ADM_PUESTO	ADM_DEPARTAMENTO	ADM_PROVINCIA	ADM_CANTON	ADM_PARROQUIA	ADM_SECTOR	CAT_RESOLUCION	CAT_TIPOENTIDAD	CAT_CARGO	CAT_TIPOTRAMITE	CAT_PRIORIDAD	CAT_ESTADOTRAMITE	TRA_DOCUMENTO	CAT_TIPODOCUMENTO
Solicitar_Clave	M																		
Adicionar_Persona	C																		
Modificar_Persona	M																		
Eliminar_Persona	E																		
Consultar_Persona	M																		
Adicionar_Cargo														C					
Modificar_Cargo														M					
Eliminar_Cargo														E					
Consultar_Cargo														M					
Adicionar_EstadoTramite																	C		
Modificar_EstadoTramite																	M		
Eliminar_EstadoTramite																	E		
Consultar_EstadoTramite																	M		
Adicionar_Prioridad																C			
Modificar_Prioridad																M			
Eliminar_Prioridad																E			
Consultar_Prioridad																M			

Adicionar_Resolucion												C								
Modificar_Resolucion												M								
Eliminar_Resolucion												E								
Consultar_Resolucion												M								
Adicionar_Documento		M																	C	M
Modificar_Documento		M																	M	M
Consultar_Documento		M																	M	M
Adicionar_TipoDocumento																				C
Modificar_TipoDocumento																				M
Consultar_TipoDocumento																				M
Adicionar_TipoEntidad													C							
Modificar_TipoEntidad													M							
Eliminar_TipoEntidad													E							
Consultar_TipoEntidad													M							
Adicionar_TipoTramite															C					
Modificar_TipoTramite															M					
Eliminar_TipoTramite															E					
Consultar_TipoTramite															M					
Adicionar_Departamento							C													
Modificar_Departamento							M													
Eliminar_Departamento							E													
Consultar_Departamento							M													
Desplegar_EstructuraDepartamental							M													
Adicionar_PuestoTrabajo						C	M													
Modificar_PuestoTrabajo						M	M													
Eliminar_PuestoTrabajo						E														

Adicionar_Funcion	M				C	M													
Modificar_Funcion	M				M	M													
Eliminar_Funcion					E														
Listar_FuncionActivo	M				M	M													
Registrar_Tramite	M		C	M										M	M	M	M		
Modificar_Tramite	M		M	M										M	M	M	M		
Consultar_TramiteRevision	M		M	M										M	M	M	M		
Mostrar_NumeroTramitesPendientes			M																
Ingresar_SeguimientoTramite		C	M		M	M						M							
Modificar_SeguimientoTramite		M	M		M	M						M							
Eliminar_SeguimientoTramite		E																	
Consultar_SeguimientoTramite		M	M		M	M						M							
Adicionar_Parroquia									M	C									
Modificar_Parroquia									M	M									
Eliminar_Parroquia										E									
Consultar_Parroquia									M	M									
Adicionar_Canton								M	C										
Modificar_Canton								M	M										
Eliminar_Canton									E										
Consultar_Canton								M	M										
Adicionar_Provincia								C											
Modificar_Provincia								M											
Eliminar_Provincia								E											
Consultar_Provincia								M											
Adicionar_Sector											C								
Modificar_Sector											M								

Eliminar_Sector											E									
Listar_Sectores											M									
Adicionar_Entidad				C						M	M		M							
Modificar_Entidad				M						M	M		M							
Eliminar_Entidad				E																
Consultar_Entidad				M						M	M		M							
Listar_DocumentoAnexos		M																	M	M
Visualizar_DocumentoAnexo																			M	
SUMATORIA	11	8	7	7	7	9	7	7	7	7	7	7	7	7	7	7	7	5	7	
TOTAL SUMATORIA OET= 138																				

- Creación de un objeto: **C**
- Modificación de un objeto: **M**
- Terminación de un objeto: **E**

TÉCNICA DE MÉTRICAS BANG PARA EL SISTEMA SIGTE

- El sistema SIGTE está fuertemente orientado a los datos, ya que implementa o gestionan una base de datos.
- Indicador principal es: Factor OB (Número de objetos del modelo de datos del sistema SIGTE).

- OB = 19

Factores de corrección de (pesos) para los objetos de un sistema, en función de sus interrelaciones con otros.

RE_i	COBI
1	1.0
2	2.3
3	4.0
4	5.8
5	7.8
6	9.8

OBJETO(entidad)	RE	COBI
TRA_PERSONA	2	2.3
TRA_SEGUIMIENTO	5	7.8
TRA_TRAMITE	7	11.8
TRA_ENTIDAD	4	5.8
ADM_FUNCION	3	4.0
ADM_PUESTO	3	4.0

ADM_DEPARTAMENTO	1	1.0
ADM_PROVINCIA	1	1.0
ADM_CANTON	2	2.3
ADM_PARROQUIA	2	2.3
ADM_SECTOR	1	1.0
CAT_RESOLUCION	1	1.0
CAT_TIPOENTIDAD	1	1.0
CAT_CARGO	1	1.0
CAT_TIPOTRAMITE	1	1.0
CAT_PRIORIDAD	1	1.0
CAT_ESTADOTRAMITE	1	1.0
TRA_DOCUMENTO	2	2.3
CAT_TIPODOCUMENTO	1	1.0

- **COB** (Corrected Object) es el **tamaño funcional total** de la aplicación medida.

$$COB = \sum_{i} COBI_i$$

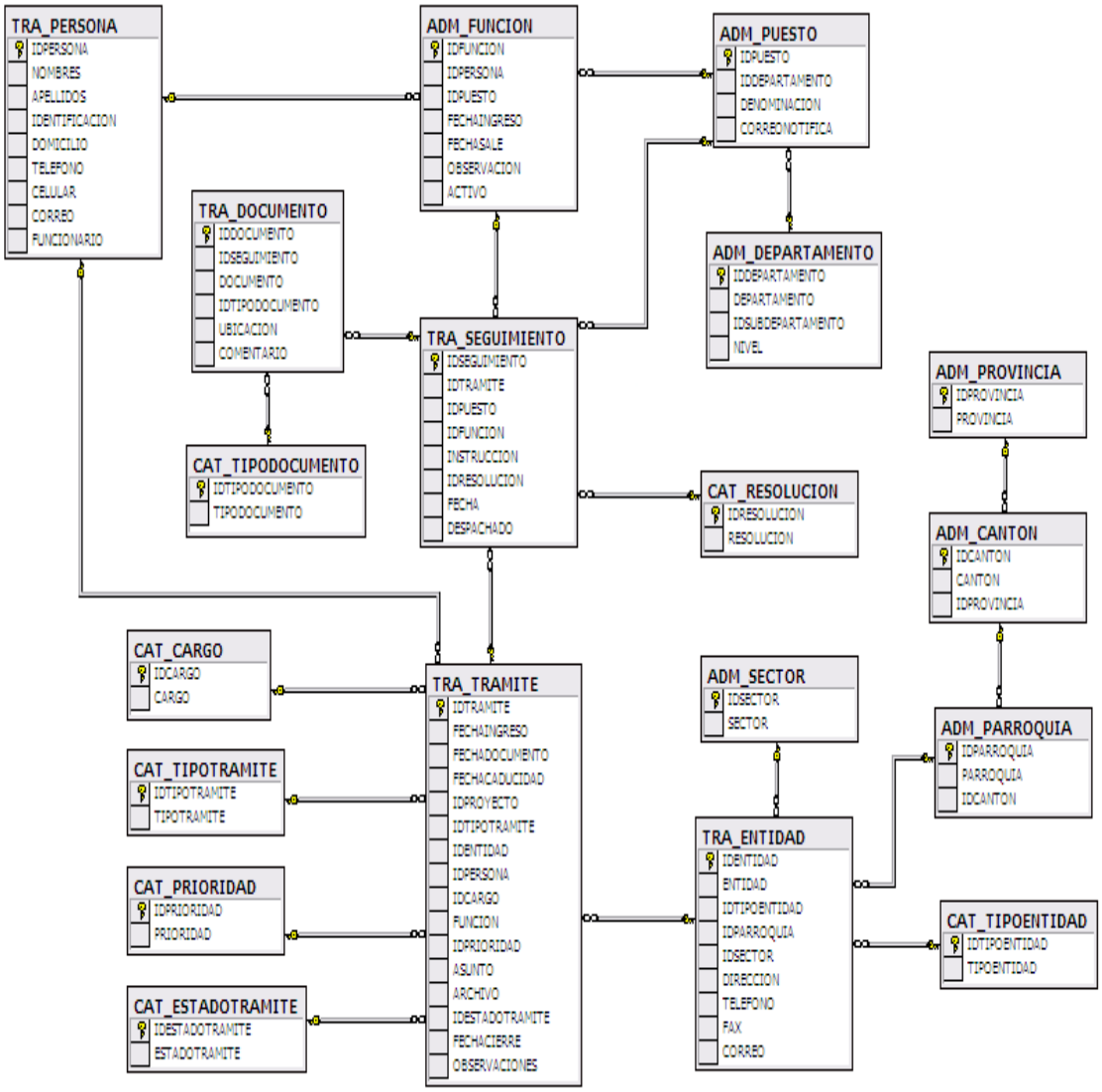
- COB = 52.6

ANEXO D
SIGTE: Sistema de Gestión de Trámites

NUEVA PROPUESTA CÁLCULO DE PUNTOS DE FUNCIÓN

METODO PARA CALCULAR LA FUNCIONALIDAD DEL SISTEMA

1) MODELO ENTIDAD RELACION SISTEMA SIGTE



ANÁLISIS DE LOS ELEMENTOS DE FUNCIÓN DEL PROYECTO SIGTE

Elemento de Función	Funciones	Tablas Afectadas FTRs	Tipos de Elementos de Datos DETs	Complejidad ILFs
Entrada Externa	Adicionar_Persona	1	10	Baja
Entrada Externa	Modificar_Persona	1	10	Baja
Entrada Externa	Eliminar_Persona	1	10	Baja
Consulta Externa	Consultar_Persona	1	10	Baja
Entrada Externa	Adicionar_Cargo	1	3	Baja
Entrada Externa	Modificar_Cargo	1	3	Baja
Entrada Externa	Eliminar_Cargo	1	3	Baja
Consulta Externa	Consultar_Cargo	1	3	Baja
Entrada Externa	Adicionar_EstadoTramite	1	3	Baja
Entrada Externa	Modificar_EstadoTramite	1	3	Baja
Entrada Externa	Eliminar_EstadoTramite	1	3	Baja
Consulta Externa	Consultar_EstadoTramite	1	3	Baja
Entrada Externa	Adicionar_Prioridad	1	3	Baja
Entrada Externa	Modificar_Prioridad	1	3	Baja
Entrada Externa	Eliminar_Prioridad	1	3	Baja
Consulta Externa	Consultar_Prioridad	1	3	Baja
Entrada Externa	Adicionar_Resolucion	1	3	Baja
Entrada Externa	Modificar_Resolucion	1	3	Baja
Entrada Externa	Eliminar_Resolucion	1	3	Baja
Consulta Externa	Consultar_Resolucion	1	3	Baja
Entrada Externa	Adicionar_Documento	3	7	Alta

Entrada Externa	Modificar_Documento	3	7	Alta
Entrada Externa	Eliminar_Documento	3	7	Alta
Consulta Externa	Consultar_Documento	3	7	Media
Entrada Externa	Adicionar_TipoDocumento	1	3	Baja
Entrada Externa	Modificar_TipoDocumento	1	3	Baja
Entrada Externa	Eliminar_TipoDocumento	1	3	Baja
Consulta Externa	Consultar_TipoDocumento	1	3	Baja
Entrada Externa	Adicionar_TipoEntidad	1	3	Baja
Entrada Externa	Modificar_TipoEntidad	1	3	Baja
Entrada Externa	Eliminar_TipoEntidad	1	3	Baja
Consulta Externa	Consultar_TipoEntidad	1	3	Baja
Entrada Externa	Adicionar_TipoTramite	1	3	Baja
Entrada Externa	Modificar_TipoTramite	1	3	Baja
Entrada Externa	Eliminar_TipoTramite	1	3	Baja
Consulta Externa	Consultar_TipoTramite	1	3	Baja
Entrada Externa	Adicionar_Departamento	1	5	Baja
Entrada Externa	Modificar_Departamento	1	5	Baja
Entrada Externa	Eliminar_Departamento	1	5	Baja
Consulta Externa	Consultar_Departamento	1	5	Baja
Entrada Externa	Adicionar_PuestoTrabajo	2	5	Baja
Entrada Externa	Modificar_PuestoTrabajo	2	5	Baja
Entrada Externa	Eliminar_PuestoTrabajo	2	5	Baja
Consulta Externa	Consultar_PuestoTrabajo	2	5	Baja
Entrada Externa	Adicionar_Funcion	3	8	Alta

Entrada Externa	Modificar_Funcion	3	8	Alta
Entrada Externa	Eliminar_Funcion	3	8	Alta
Consulta Externa	Consultar_Funcion	3	8	Media
Entrada Externa	Registrar_Tramite	7	17	Alta
Entrada Externa	Modificar_Tramite	7	17	Alta
Entrada Externa	Eliminar_Tramite	7	17	Alta
Consulta Externa	Consultar_Tramite	7	17	Alta
Entrada Externa	Ingresar_SeguimientoTramite	5	9	Alta
Entrada Externa	Modificar_SeguimientoTramite	5	9	Alta
Entrada Externa	Eliminar_SeguimientoTramite	5	9	Alta
Consulta Externa	Consultar_SeguimientoTramite	5	9	Alta
Entrada Externa	Adicionar_Parroquia	2	4	Baja
Entrada Externa	Modificar_Parroquia	2	4	Baja
Entrada Externa	Eliminar_Parroquia	2	4	Baja
Consulta Externa	Consultar_Parroquia	2	4	Baja
Entrada Externa	Adicionar_Canton	2	4	Baja
Entrada Externa	Modificar_Canton	2	4	Baja
Entrada Externa	Eliminar_Canton	2	4	Baja
Consulta Externa	Consultar_Canton	2	4	Baja
Entrada Externa	Adicionar_Provincia	1	3	Baja
Entrada Externa	Modificar_Provincia	1	3	Baja
Entrada Externa	Eliminar_Provincia	1	3	Baja
Consulta Externa	Consultar_Provincia	1	3	Baja
Entrada Externa	Adicionar_Sector	1	3	Baja

Entrada Externa	Modificar_Sector	1	3	Baja
Entrada Externa	Eliminar_Sector	1	3	Baja
Consulta Externa	Consultar_Sector	1	3	Baja
Entrada Externa	Adicionar_Entidad	4	10	Alta
Entrada Externa	Modificar_Entidad	4	10	Alta
Entrada Externa	Eliminar_Entidad	4	10	Alta
Consulta Externa	Consultar_Entidad	4	10	Alta

Archivos Lógicos ILFs	**Registros Lógicos**	**Tipos de Elementos de Datos DETs**	**Complejidad ILFs**
TRA_PERSONA	1	9	Baja
TRA_SEGUIMIENTO	1	8	Baja
TRA_TRAMITE	1	16	Baja
TRA_ENTIDAD	1	9	Baja
ADM_FUNCION	1	7	Baja
ADM_PUESTO	1	4	Baja
ADM_DEPARTAMENTO	1	4	Baja
ADM_PROVINCIA	1	2	Baja
ADM_CANTON	1	3	Baja
ADM_PARROQUIA	1	3	Baja
ADM_SECTOR	1	2	Baja
CAT_RESOLUCION	1	2	Baja
CAT_TIPOENTIDAD	1	2	Baja
CAT_CARGO	1	2	Baja
CAT_TIPOTRAMITE	1	2	Baja
CAT_PRIORIDAD	1	2	Baja

CAT_ESTADOTRAMITE	1	2	Baja
TRA_DOCUMENTO	1	6	Baja
CAT_TIPODOCUMENTO	1	2	Baja

ENTRADAS EXTERNAS

Archivos referenciados	Elementos de Datos		
	1-4	5-15	>15
0-1	Baja(30)	Baja(3)	Media
2	Baja(9)	Media	Alta
3 ó más	Media	Alta(12)	Alta(3)

CONSULTAS EXTERNAS

Archivos referenciados	Elementos de Datos		
	1-5	6-19	>19
0-1	Baja(10)	Baja(1)	Media
2-3	Baja(3)	Media(2)	Alta
>3	Media	Alta(3)	Alta

ARCHIVOS LÓGICOS INTERNOS

Tipos de Registros	Elementos de Datos		
	1-19	20-50	>50
1	Baja(19)	Baja	Media
2-5	Baja	Media	Alta
>5	Media	Alta	Alta

PUNTOS DE FUNCION SIN AJUSTAR (PFSA)

Punto de Función	Complejidad			
	Alta	**Media**	**Baja**	**Total**
Entradas Externas	6***15**	4	3***42**	216
Salidas Externas	7	5	4	0
Consultas Externas	6***3**	4***2**	3***14**	68
Archivos Lógicos Internos	15	10	7***19**	133

Interfases Externas	10	7	5	0
			PFSA	**417**

CARACTERISTICAS GENERALES DEL SISTEMA

Valor	Significado
0	Sin influencia, factor no presente
1	Influencia insignificante, muy baja
2	Influencia moderada o baja
3	Influencia media, normal
4	Influencia alta, significativa
5	Influencia muy alta, esencial

Características Generales del Sistema		Complejidad	Comentario
1	Comunicación de datos	3	Comunicación de datos moderada.
2	Procesamiento de datos distribuido	1	Sistema distribuido con conexión remota o directa a la red.
3	Rendimiento	2	Tiempos de respuesta que no superen los 10 segundos en redes extendidas y menores que 3 segundos sobre redes locales.
4	Uso de hardware existente	1	No se ha detallado.
5	Transacciones	3	Moderado manejo de transacciones.
6	Entrada de datos interactiva	4	Manejo de browser de Internet para la entrada de datos.
7	Eficiencia	4	Si existe eficiencia del sistema.
8	Actualizaciones on-line	4	Manejo de Internet para ejecutar transacciones.
9	Complejidad de procesamiento	1	Hay pocos cálculos en el sistema.
10	Reusabilidad	0	No se requiere que el código sea reutilizable
11	Facilidad de conversión e instalación	3	Los usuarios disponen de terminales con la configuración mínima requerida por los navegadores web.
12	Facilidad de operación	3	Operación normal.
13	Múltiples instalaciones	1	Se requiere escasas instalaciones del sistema en todos los equipos.
14	Facilidad de mantenimiento	3	Se requiere un costo moderado de mantenimiento.
Valor de Complejidad VAF		**33**	

CÁLCULOS DE PUNTOS DE FUNCION AJUSTADO:

PFSA= Punto de Función sin ajustar= 417

PFA= PFSA *((VAF*0.01)+0.65)

VAF= Grado Total de Complejidad= 33

- PFA= 417((33*0.01)+0.65)= 408.66

TRANSFORMACIÓN PF A SLOC

Size = 32 (VB) x 417 PFSA = 13344 SLOC = **13.344 KSLOC**

CALCULO DEL ESFUERZO FUNCIONAL DEL SISTEMA

CÁLCULO DE ESFUERZO NOMINAL

$$PM_{nominal} = A \times (Size)^{B}$$

AJUSTE DE FACTORES DE ESCALA (B)

FÓRMULA:

$$B = 1.01 + 0.01 \times \sum_{j=1}^{5} Wj$$

VARIABLE	DESCRIPCION	PODERACIÓN	VALOR
PREC	El sistema es familiar.	Nominal	3.72
FLEX	La flexibilidad de desarrollo es moderada.	Nominal	3.04
RESL	La arquitectura es normal y los riesgos generalmente se mitigan.	Nominal	4.24
TEAM	La interacción del equipo es normal.	Nominal	3.29
PMAT	La madurez del proceso software es normal.	Nominal	4.68
		TOTAL	**1.20**

$PM_{nominal} = 2.94 \times (13.344)^{1.20}$

$PM_{nominal} = 65.87$ Meses-Persona

CÀLCULO DE ESFUERZO AJUSTADO

$$PM_{ajustado} = PM_{nominal} \times \Pi(ME_i)$$

AJUSTE DE DRIVERS DE COSTO

FÓRMULA:

$$\Pi(ME_i)$$

MULTIPLICADOR	DESCRIPCION	PONDERACIÓN	VALOR
PERS	Se tiene analistas y programadores con gran eficiencia y buen trabajo en equipo. Dedicación full-time.	Nominal	1
RCPX	La exigencia de confiabilidad, documentación y volumen de datos son moderadas, el producto no tiene mucha complejidad.	Nominal	1
RUSE	Se pretende reutilizar nada.	Nominal	1
PDIF	Existen restricciones al tiempo de respuesta. No se especifica el hardware a utilizar, ni la plataforma de soporte.	Nominal	1
PREX	Todos los analistas y programadores tienen una moderada experiencia de desarrollo de la aplicación.	Nominal	1
SCED	Se requiere terminar el proyecto en el tiempo estimado.	Nominal	1
FCIL	Se requiere herramienta CASE para desarrollo Web. La comunicación de datos debe ser distribuido con conexión a Internet.	Nominal	1
		TOTAL	**1**

$PM_{ajustado}$= 65.87 x 1

$PM_{ajustado}$= 65.87 Meses-persona

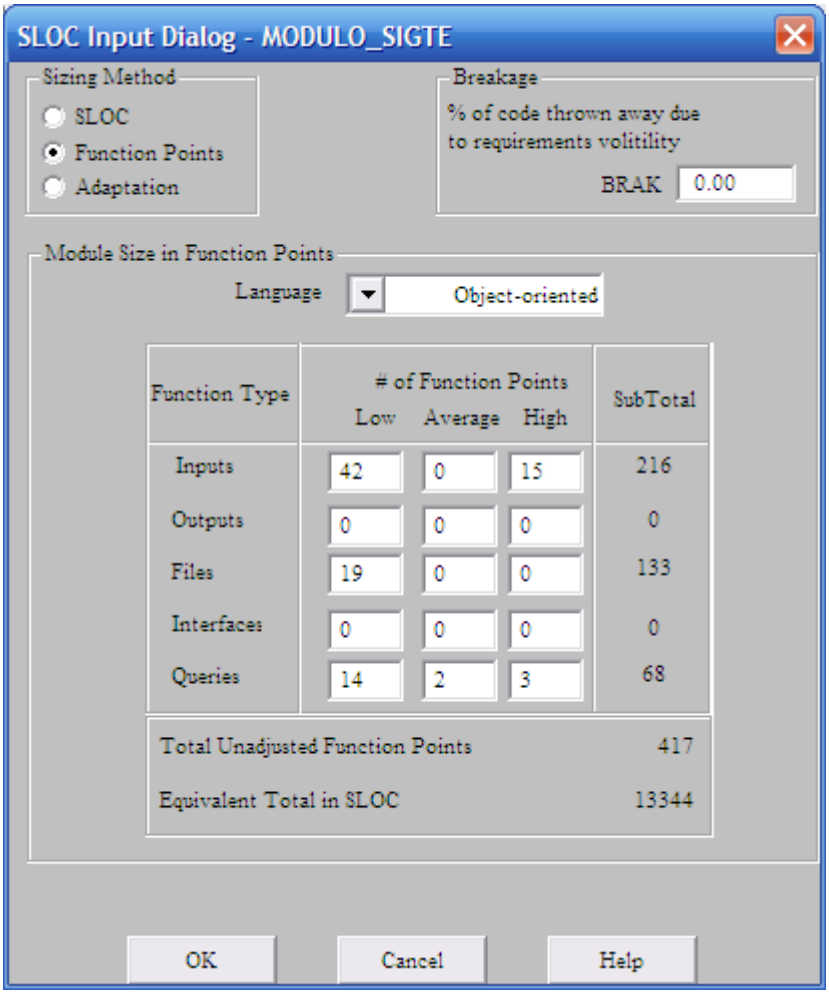
CÁLCULO DEL TIEMPO DE DESARROLLO

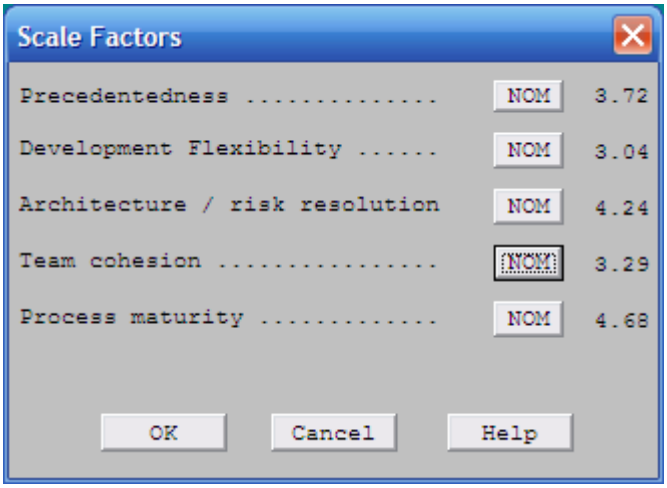
$$TDEV = [3.67 \times PM^{(0.28+0.2\times(B-1.01))}] \times \frac{SCED\%}{100}$$

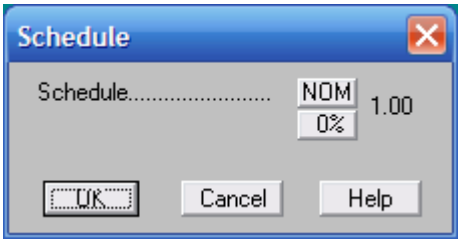
TDEV= [3.67 X 65.87 $^{(0.278+0.2x(1.20-1.01))}$] x 1

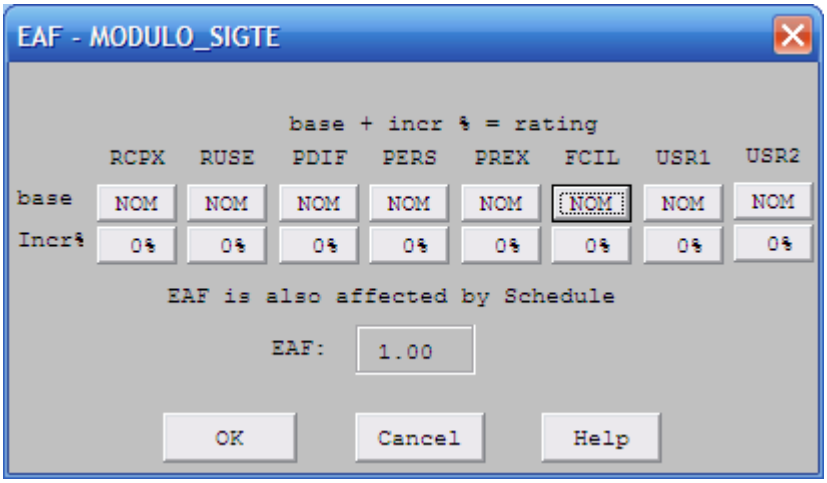
TDEV= [3.67 X 65.87 $^{0.316}$] x 1 **= 13.78 Meses**

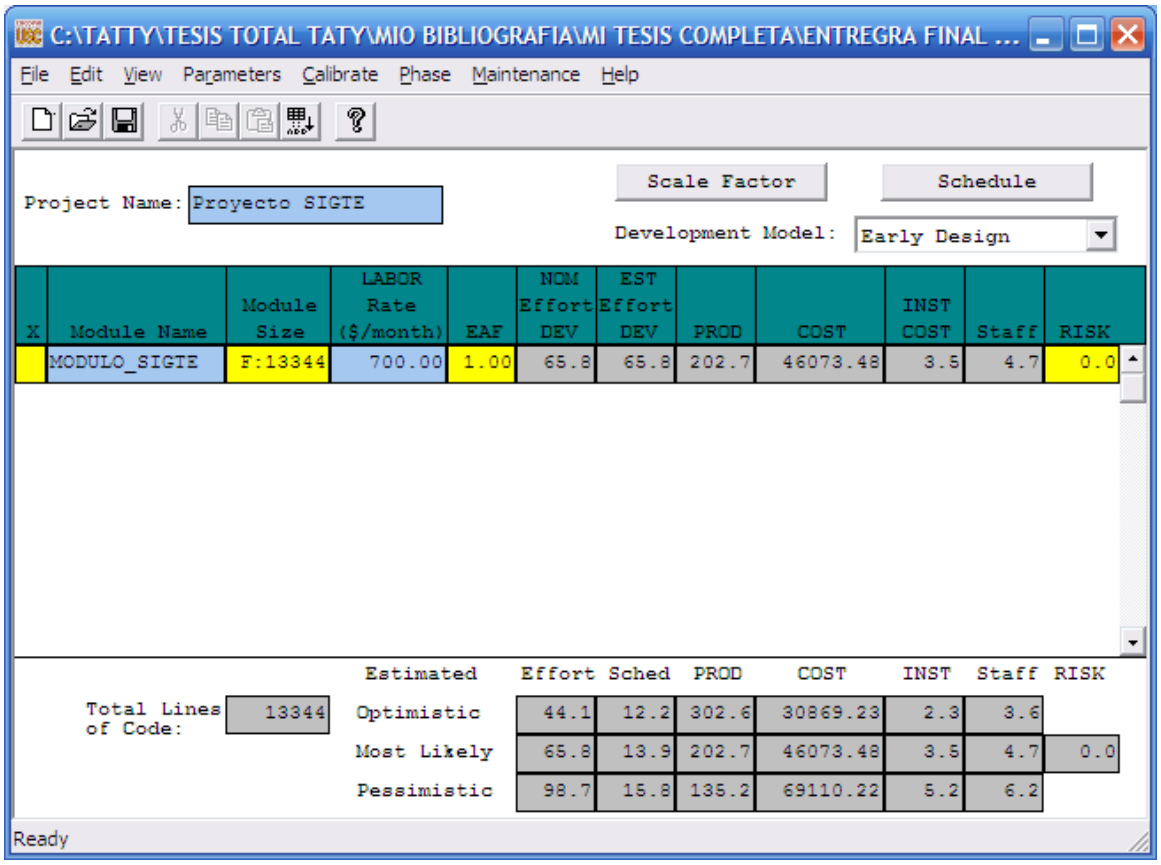
CÁLCULO MEDIANTE HERRAMIENTA COCOMO



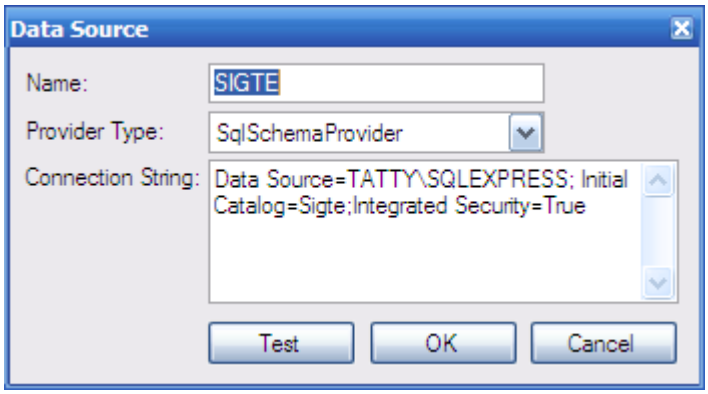


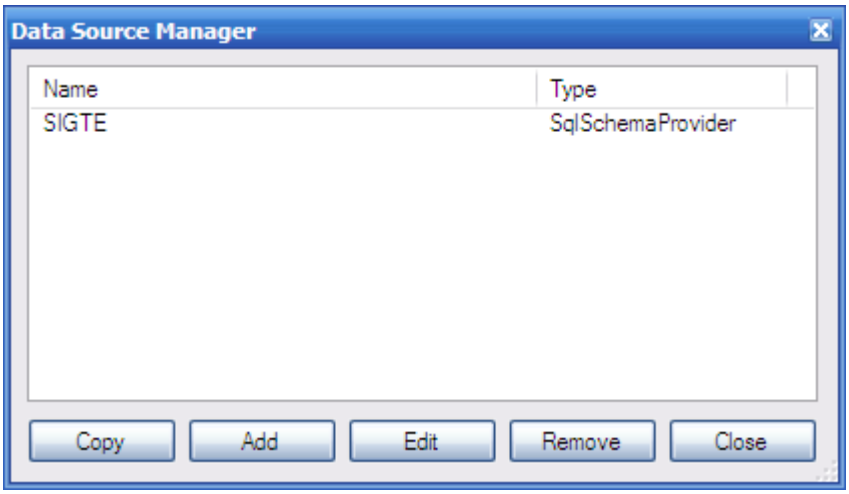

Schedule
Schedule.....
NOM
0%
1.00
OK
Cancel
Help

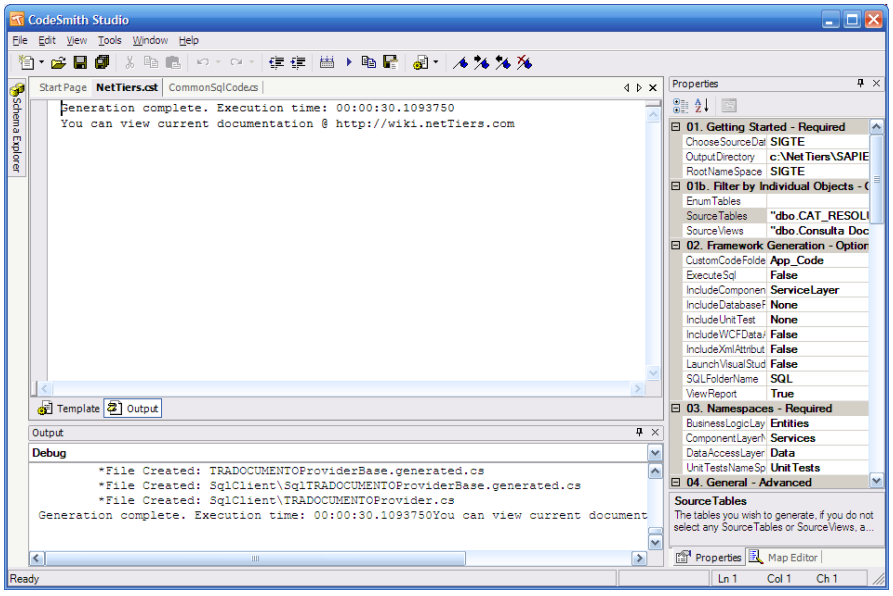

EAF - MODULO_SIGTE
base + incr % = rating
RCPX RUSE PDIF PERS PREX FCIL USR1 USR2
base NOM NOM NOM NOM NOM NOM NOM NOM
Incr% 0% 0% 0% 0% 0% 0% 0% 0%
EAF is also affected by Schedule
EAF: 1.00
OK
Cancel
Help


C:\TATTY\TESIS TOTAL TATY\MIO BIBLIOGRAFIA\MI TESIS COMPLETA\ENTREGRA FINAL ...
File Edit View Parameters Calibrate Phase Maintenance Help
Project Name: Proyecto SIGTE
Scale Factor
Schedule
Development Model: Early Design
X Module Name Module Size LABOR Rate ($/month) EAF NOM Effort DEV EST Effort DEV PROD COST INST COST Staff RISK
MODULO_SIGTE F:13344 700.00 1.00 65.8 65.8 202.7 46073.48 3.5 4.7 0.0
Estimated Effort Sched PROD COST INST Staff RISK
Total Lines of Code: 13344
Optimistic 44.1 12.2 302.6 30869.23 2.3 3.6
Most Likely 65.8 13.9 202.7 46073.48 3.5 4.7 0.0
Pessimistic 98.7 15.8 135.2 69110.22 5.2 6.2
Ready

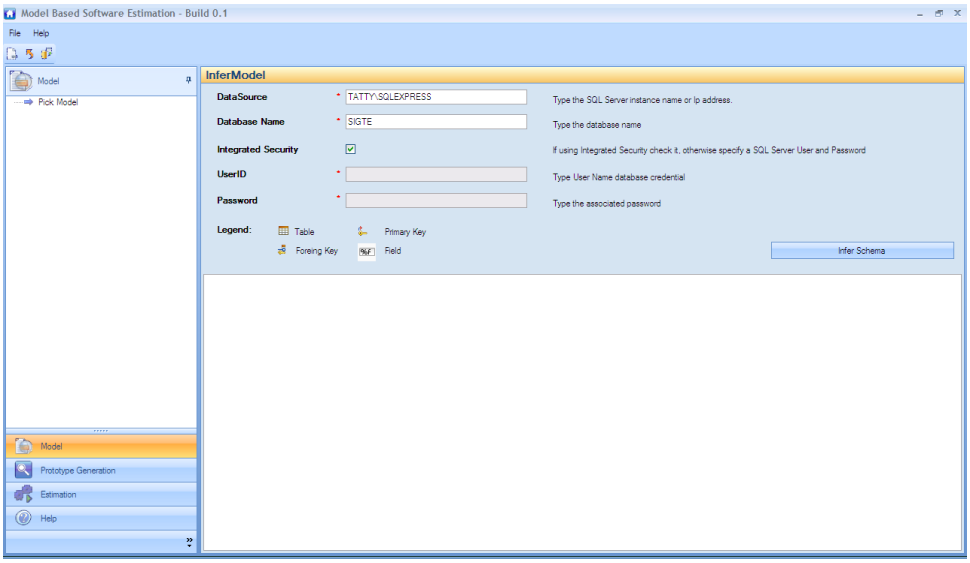
METODO PARA CALCULAR LA ARQUITECTURA DEL SISTEMA – HERRAMIENTA CODESMITH STUDIO

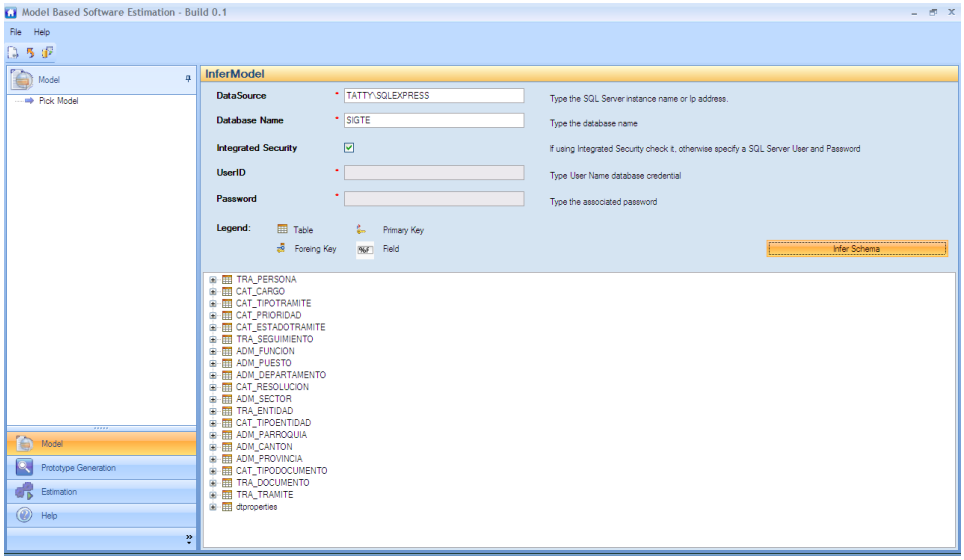


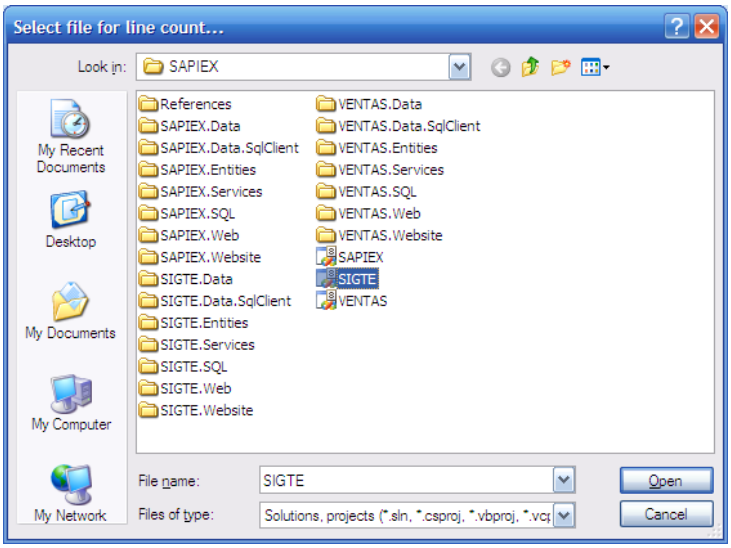


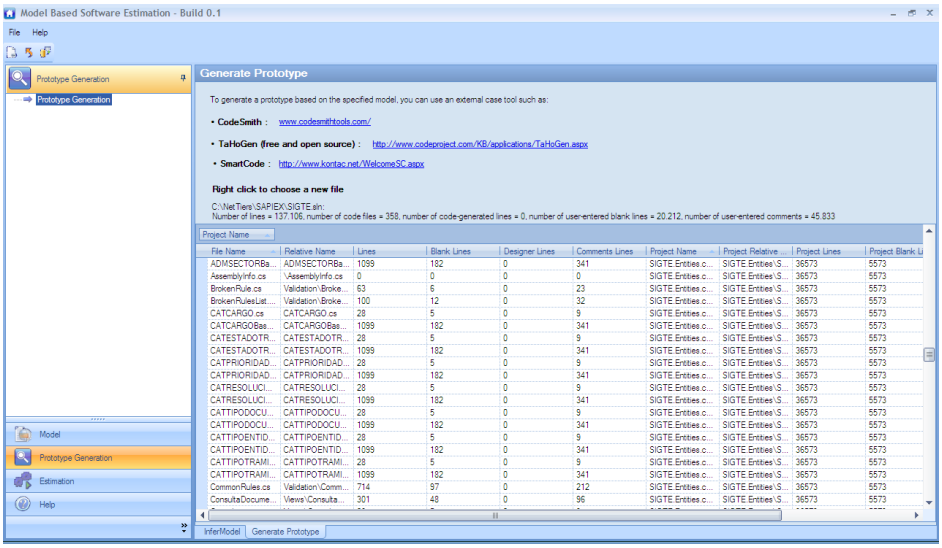


USO DE LA HERRAMIENTA MPE









TLOC= 70703

ALOC= TLOC-BLOC

ALOC= 70703-13344= 57359 SLOC

CALCULO DEL ESFUERZO DE LA ARQUITECTURA DEL SISTEMA

CÁLCULO DE ESFUERZO NOMINAL

$$PM_{nominal} = A \times (Size)^{B}$$

AJUSTE DE FACTORES DE ESCALA (B)

FÓRMULA:

$$B = 1.01 + 0.01 \times \sum_{j=1}^{5} Wj$$

VARIABLE	DESCRIPCION	PODERACIÓN	VALOR
PREC	El sistema es familiar.	Nominal	3.72
FLEX	La flexibilidad de desarrollo es moderada.	Nominal	3.04
RESL	La arquitectura es normal y los riesgos generalmente se mitigan.	Nominal	4.24
TEAM	La interacción del equipo es normal.	Nominal	3.29
PMAT	La madurez del proceso	Nominal	4.68

	software es normal.		
		TOTAL	**1.20**

$PM_{nominal}$= 2.94 x $(57.359)^{1.20}$

$PM_{nominal}$= 379.02 Meses-Persona

CÀLCULO DE ESFUERZO AJUSTADO

$$PM_{ajustado} = PM_{nominal} \times \Pi(ME_i)$$

AJUSTE DE DRIVERS DE COSTO

FÓRMULA:

$$\Pi(ME_i)$$

MULTIPLICADOR	DESCRIPCION	PONDERACIÓN	VALOR
PERS	Se tiene analistas y programadores con gran eficiencia y buen trabajo en equipo. Dedicación full-time.	Nominal	1
RCPX	La exigencia de confiabilidad, documentación y volumen de datos son moderadas, el producto no tiene mucha complejidad.	Nominal	1
RUSE	Se pretende reutilizar nada.	Nominal	1
PDIF	Existen restricciones al tiempo de respuesta. No se especifica el hardware a utilizar, ni la plataforma de soporte.	Nominal	1
PREX	Todos los analistas y programadores tienen una moderada experiencia de desarrollo de la aplicación.	Nominal	1
SCED	Se requiere terminar el proyecto en el tiempo estimado.	Nominal	1
FCIL	Se requiere herramienta CASE para desarrollo Web. La comunicación de datos debe ser distribuido con conexión a Internet.	Nominal	1
		TOTAL	**1**

$PM_{ajustado}$= 379.02 x 1

$PM_{ajustado}$= 379.02 Meses-persona

CÁLCULO DEL TIEMPO DE DESARROLLO

$$TDEV = [3.67 \times PM^{(0.28+0.2\times(B-1.01))}] \times \frac{SCED\%}{100}$$

TDEV= [3.67 X $379.02^{(0.278+0.2x(1.20-1.01))}$] x 1

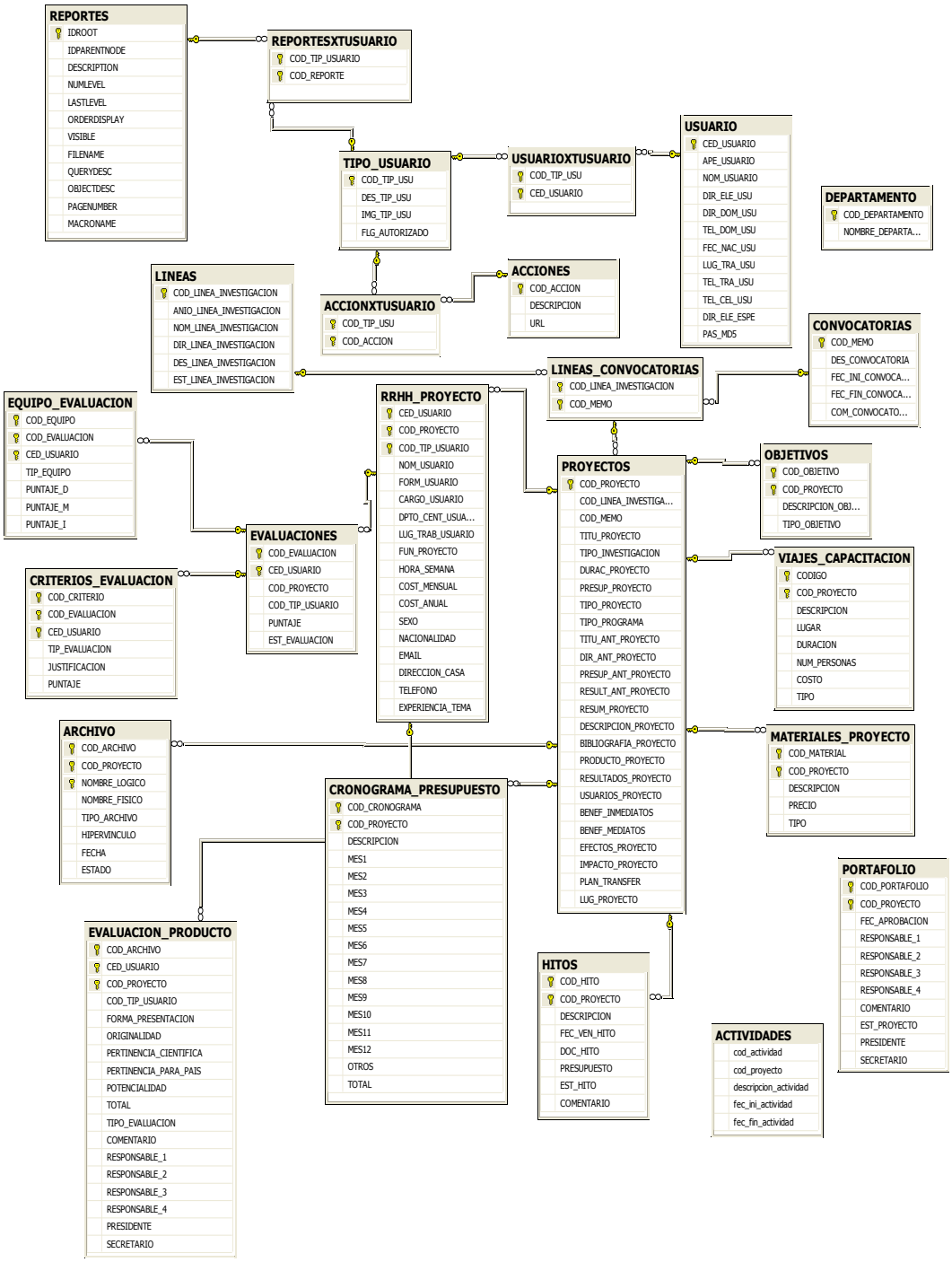
TDEV= [3.67 X $379.02^{0.316}$] x 1 **= 23.96 Meses**

ANEXO E
SAPIV: Sistema de Administración de Proyectos de Investigación

NUEVA PROPUESTA CÁLCULO DE PUNTOS DE FUNCIÓN

METODO PARA CALCULAR LA FUNCIONALIDAD DEL SISTEMA

MODELO ENTIDAD RELACION SISTEMA SAPIV



ANÁLISIS DE LOS ELEMENTOS DE FUNCIÓN DEL PROYECTO SAPIEX

Elemento de Función	Funciones	Tablas Afectadas FTRs	Tipos de Elementos de Datos DETs	Complejidad ILFs
Entrada Externa	Adicionar_Reporte	1	13	Baja
Entrada Externa	Modificar_Reporte	1	13	Baja
Entrada Externa	Eliminar_Reporte	1	13	Baja
Consulta Externa	Consultar_Reporte	1	13	Baja
Entrada Externa	Adicionar_ReportesxtUsuario	3	3	Media
Entrada Externa	Modificar_ ReportesxtUsuario	3	3	Media
Entrada Externa	Eliminar_ ReportesxtUsuario	3	3	Media
Consulta Externa	Consultar_ ReportesxtUsuario	3	3	Media
Entrada Externa	Adicionar_TipoUsuario	1	5	Baja
Entrada Externa	Modificar_ TipoUsuario	1	5	Baja
Entrada Externa	Eliminar_ TipoUsuario	1	5	Baja
Consulta Externa	Consultar_ TipoUsuario	1	5	Baja
Entrada Externa	Adicionar_UsuarioxtUsuario	3	3	Media
Entrada Externa	Modificar_ UsuarioxtUsuario	3	3	Media
Entrada Externa	Eliminar_ UsuarioxtUsuario	3	3	Media
Consulta Externa	Consultar_ UsuarioxtUsuario	3	3	Media
Entrada Externa	Adicionar_Usuario	1	13	Baja
Entrada Externa	Modificar_Usuario	1	13	Baja
Entrada Externa	Eliminar_Usuario	1	13	Baja
Consulta Externa	Consultar_Usuario	1	13	Baja
Entrada Externa	Adicionar_Acciones	1	4	Baja

Entrada Externa	Modificar_ Acciones	1	4	Baja
Entrada Externa	Eliminar_ Acciones	1	4	Baja
Consulta Externa	Consultar_ Acciones	1	4	Baja
Entrada Externa	Adicionar_AccionxtUsuario	3	3	Media
Entrada Externa	Modificar_ AccionxtUsuario	3	3	Media
Entrada Externa	Eliminar_ AccionxtUsuario	3	3	Media
Consulta Externa	Consultar_ AccionxtUsuario	3	3	Media
Entrada Externa	Adicionar_Departamento	1	3	Baja
Entrada Externa	Modificar_ Departamento	1	3	Baja
Entrada Externa	Eliminar_ Departamento	1	3	Baja
Consulta Externa	Consultar_ Departamento	1	3	Baja
Entrada Externa	Adicionar_Lineas	1	7	Baja
Entrada Externa	Modificar_Lineas	1	7	Baja
Entrada Externa	Eliminar_Lineas	1	7	Baja
Consulta Externa	Consultar_Lineas	1	7	Baja
Entrada Externa	Adicionar_LineasConvocatorias	3	3	Media
Entrada Externa	Modificar_LineasConvocatorias	3	3	Media
Entrada Externa	Eliminar_LineasConvocatorias	3	3	Media
Consulta Externa	Consultar_LineasConvocatorias	3	3	Media
Entrada Externa	Adicionar_Convocatorias	1	6	Baja
Entrada Externa	Modificar_Convocatorias	1	6	Baja
Entrada Externa	Eliminar_Convocatorias	1	6	Baja
Consulta Externa	Consultar_Convocatorias	1	6	Baja
Entrada Externa	Adicionar_ Evaluaciones	2	7	Media

Entrada Externa	Modificar_ Evaluaciones	2	7	Media
Entrada Externa	Eliminar_ Evaluaciones	2	7	Media
Consulta Externa	Consultar_ Evaluaciones	2	7	Media
Entrada Externa	Adicionar_EquipoEvaluacion	2	8	Media
Entrada Externa	Modificar_ EquipoEvaluacion	2	8	Media
Entrada Externa	Eliminar_ EquipoEvaluacion	2	8	Media
Consulta Externa	Consultar_ EquipoEvaluacion	2	8	Media
Entrada Externa	Adicionar_CriteriosEvaluacion	2	7	Media
Entrada Externa	Modificar_ CriteriosEvaluacion	2	7	Media
Entrada Externa	Eliminar_ CriteriosEvaluacion	2	7	Media
Consulta Externa	Consultar_ CriteriosEvaluacion	2	7	Media
Entrada Externa	Registrar_RRHHProyecto	2	19	Alta
Entrada Externa	Modificar_ RRHHProyecto	2	19	Alta
Entrada Externa	Eliminar_ RRHHProyecto	2	19	Alta
Consulta Externa	Consultar_ RRHHProyecto	2	19	Media
Entrada Externa	Adicionar_Proyectos	2	26	Alta
Entrada Externa	Modificar_ Proyectos	2	26	Alta
Entrada Externa	Eliminar_ Proyectos	2	26	Alta
Consulta Externa	Consultar_ Proyectos	2	26	Alta
Entrada Externa	Adicionar_Objetivos	2	5	Media
Entrada Externa	Modificar_ Objetivos	2	5	Media
Entrada Externa	Eliminar_ Objetivos	2	5	Media
Consulta Externa	Consultar_ Objetivos	2	5	Baja
Entrada Externa	Adicionar_ViajesCapacitacion	2	9	Media

Entrada Externa	Modificar_ ViajesCapacitacion	2	9	Media
Entrada Externa	Eliminar_ ViajesCapacitacion	2	9	Media
Consulta Externa	Consultar_ ViajesCapacitacion	2	9	Media
Entrada Externa	Adicionar_MaterialesProyecto	2	6	Media
Entrada Externa	Modificar_ MaterialesProyecto	2	6	Media
Entrada Externa	Eliminar_ MaterialesProyecto	2	6	Media
Consulta Externa	Consultar_ MaterialesProyecto	2	6	Media
Entrada Externa	Adicionar_Portafolio	1	12	Baja
Entrada Externa	Modificar_ Portafolio	1	12	Baja
Entrada Externa	Eliminar_ Portafolio	1	12	Baja
Consulta Externa	Consultar_ Portafolio	1	12	Baja
Entrada Externa	Adicionar_Actividades	1	6	Baja
Entrada Externa	Modificar_ Actividades	1	6	Baja
Entrada Externa	Eliminar_ Actividades	1	6	Baja
Consulta Externa	Consultar_ Actividades	1	6	Baja
Entrada Externa	Adicionar_Hitos	2	9	Media
Entrada Externa	Modificar_ Hitos	2	9	Media
Entrada Externa	Eliminar_ Hitos	2	9	Media
Consulta Externa	Consultar_ Hitos	2	9	Media
Entrada Externa	Adicionar_CronogramaPresupuesto	2	18	Alta
Entrada Externa	Modificar_CronogramaPresupuesto	2	18	Alta
Entrada Externa	Eliminar_ CronogramaPresupuesto	2	18	Alta
Consulta Externa	Consultar_CronogramaPresupuesto	2	18	Media
Entrada Externa	Adicionar_EvaluacionProducto	2	19	Alta

Entrada Externa	Modificar_ EvaluacionProducto	2	19	Alta
Entrada Externa	Eliminar_ EvaluacionProducto	2	19	Alta
Consulta Externa	Consultar_ EvaluacionProducto	2	19	Media
Entrada Externa	Adicionar_Archivo	2	9	Media
Entrada Externa	Modificar_ Archivo	2	9	Media
Entrada Externa	Eliminar_ Archivo	2	9	Media
Consulta Externa	Consultar_ Archivo	2	9	Media

Archivos Lógicos ILFs	**Registros Lógicos**	**Tipos de Elementos de Datos DETs**	**Complejidad ILFs**
REPORTES	1	12	Baja
REPORTESXTUSUARIO	1	2	Baja
TIPO_USUARIO	1	4	Baja
USUARIOXTUSUARIO	1	2	Baja
USUARIO	1	12	Baja
ACCIONXTUSUARIO	1	2	Baja
ACCIONES	1	3	Baja
DEPARTAMENTO	1	2	Baja
LINEAS	1	6	Baja
EQUIPO_EVALUACION	1	7	Baja
RRHH_PROYECTO	1	18	Baja
PROYECTOS	1	25	Baja
LINEAS_CONVOCATORIAS	1	2	Baja
CONVOCATORIAS	1	5	Baja
EVALUACIONES	1	6	Baja

CRITERIOS_EVALUACION	1	6	Baja
OBJETIVOS	1	4	Baja
VIAJES_CAPACITACION	1	8	Baja
MATERIALES_PROYECTO	1	5	Baja
PORTAFOLIO	1	11	Baja
ACTIVIDADES	1	5	Baja
HITOS	1	8	Baja
CRONOGRAMA_PRESUPUESTO	1	17	Baja
EVALUACION_PRODUCTO	1	18	Baja
ARCHIVO	1	8	Baja

ENTRADAS EXTERNAS

Archivos referenciados	**Elementos de Datos**		
	1-4	5-15	>15
0-1	Baja	Baja	Media
2	Baja	Media	Alta
3 ó más	Media	Alta	Alta(3)

CONSULTAS EXTERNAS

Archivos referenciados	**Elementos de Datos**		
	1-5	6-19	>19
0-1	Baja	Baja	Media
2-3	Baja	Media	Alta
>3	Media	Alta	Alta

ARCHIVOS LÓGICOS INTERNOS

Tipos de Registros	**Elementos de Datos**		
	1-19	20-50	>50
1	Baja	Baja	Media
2-5	Baja	Media	Alta
>5	Media	Alta	Alta

PUNTOS DE FUNCION SIN AJUSTAR (PFSA)

Punto de Función	Complejidad			
	Alta	**Media**	**Baja**	**Total**
Entradas Externas	6***12**	4***36**	3***27**	297
Salidas Externas	7	5	4	0
Consultas Externas	6***1**	4***14**	3***10**	92
Archivos Lógicos Internos	15	10	7***25**	175
Interfases Externas	10	7	5	0
			PFSA	**564**

CARACTERISTICAS GENERALES DEL SISTEMA

Valor	Significado
0	Sin influencia, factor no presente
1	Influencia insignificante, muy baja
2	Influencia moderada o baja
3	Influencia media, normal
4	Influencia alta, significativa
5	Influencia muy alta, esencial

Características Generales del Sistema		Complejidad	Comentario
1	Comunicación de datos	3	Comunicación de datos moderada.
2	Procesamiento de datos distribuido	1	Sistema distribuido con conexión remota o directa a la red.
3	Rendimiento	2	Tiempos de respuesta que no superen los 10 segundos en redes extendidas y menores que 3 segundos sobre redes locales.
4	Uso de hardware existente	1	No se ha detallado.
5	Transacciones	3	Moderado manejo de transacciones.
6	Entrada de datos interactiva	4	Manejo de browser de Internet para la entrada de datos.
7	Eficiencia	4	Si existe eficiencia del sistema.
8	Actualizaciones on-line	4	Manejo de Internet para ejecutar transacciones.
9	Complejidad de procesamiento	1	Hay pocos cálculos en el sistema.
10	Reusabilidad	0	No se requiere que el código sea reutilizable
11	Facilidad de conversión e instalación	3	Los usuarios disponen de terminales con la configuración mínima requerida por los navegadores web.

12	Facilidad de operación	3	Operación normal.
13	Múltiples instalaciones	1	Se requiere escasas instalaciones del sistema en todos los equipos.
14	Facilidad de mantenimiento	3	Se requiere un costo moderado de mantenimiento.
Valor de Complejidad VAF		**33**	

CÁLCULOS DE PUNTOS DE FUNCION AJUSTADO:

PFSA= Punto de Función sin ajustar= 564

PFA= PFSA *((VAF*0.01)+0.65)

VAF= Grado Total de Complejidad= 33

- PFA= 564((33*0.01)+0.65)= 552.72

TRANSFORMACIÓN PF A SLOC

Size = 32 (VB) x 564 PFSA = 18048 SLOC = **18.048 KSLOC**

CALCULO DEL ESFUERZO FUNCIONAL DEL SISTEMA

CÁLCULO DE ESFUERZO NOMINAL

$$PM_{nominal} = A \times (Size)^{B}$$

AJUSTE DE FACTORES DE ESCALA (B)

FÓRMULA:

$$B = 1.01 + 0.01 \times \sum_{j=1}^{5} Wj$$

VARIABLE	DESCRIPCION	PODERACIÓN	VALOR
PREC	El sistema es familiar.	Nominal	3.72
FLEX	La flexibilidad de desarrollo es moderada.	Nominal	3.04
RESL	La arquitectura es normal y los riesgos generalmente se mitigan.	Nominal	4.24
TEAM	La interacción del equipo es normal.	Nominal	3.29
PMAT	La madurez del proceso software es normal.	Nominal	4.68
		TOTAL	**1.20**

$PM_{nominal} = 2.94 \times (18.048)^{1.20}$

$PM_{nominal} = 94.63$ Meses-Persona

CÀLCULO DE ESFUERZO AJUSTADO

$$PM_{ajustado} = PM_{nominal} \times \Pi(ME_i)$$

AJUSTE DE DRIVERS DE COSTO

FÓRMULA:

$$\Pi(ME_i)$$

MULTIPLICADOR	DESCRIPCION	PONDERACIÓN	VALOR
PERS	Se tiene analistas y programadores con gran eficiencia y buen trabajo en equipo. Dedicación full-time.	Nominal	1
RCPX	La exigencia de confiabilidad, documentación y volumen de datos son moderadas, el producto no tiene mucha complejidad.	Nominal	1
RUSE	Se pretende reutilizar nada.	Nominal	1
PDIF	Existen restricciones al tiempo de respuesta. No se especifica el hardware a utilizar, ni la plataforma de soporte.	Nominal	1
PREX	Todos los analistas y programadores tienen una moderada experiencia de desarrollo de la aplicación.	Nominal	1
SCED	Se requiere terminar el proyecto en el tiempo estimado.	Nominal	1
FCIL	Se requiere herramienta CASE para desarrollo Web. La comunicación de datos debe ser distribuido con conexión a Internet.	Nominal	1
		TOTAL	**1**

$PM_{ajustado} = 94.63 \times 1$

$PM_{ajustado} = 94.63$ Meses-persona

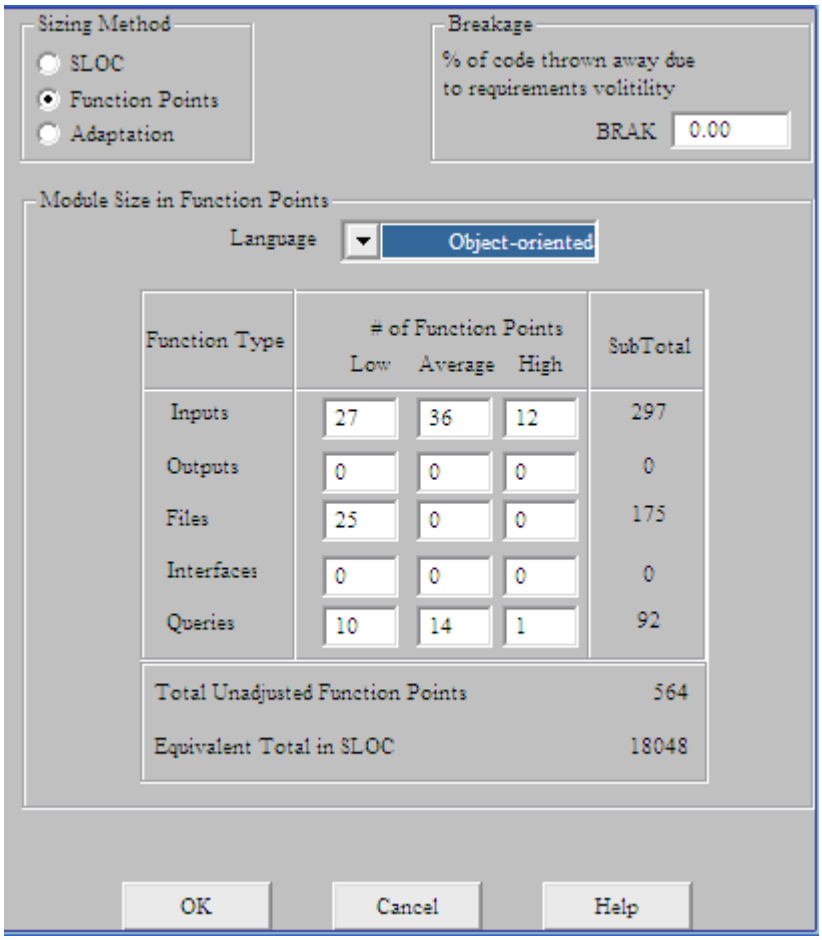
CÁLCULO DEL TIEMPO DE DESARROLLO

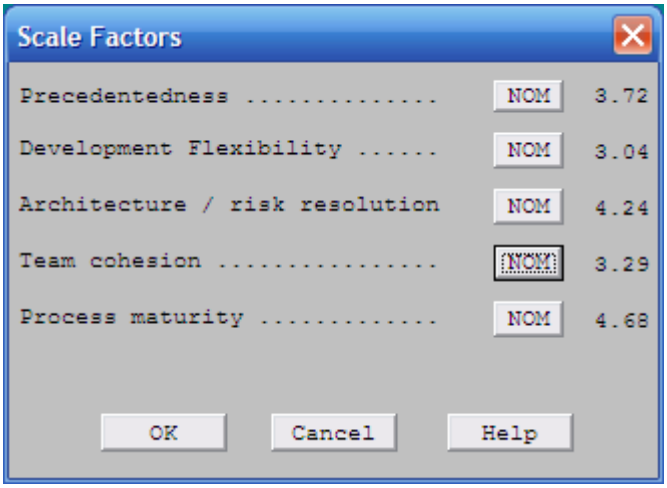
$$TDEV = [3.67 \times PM^{(0.28+0.2X(B-1.01))}] \times \frac{SCED\%}{100}$$

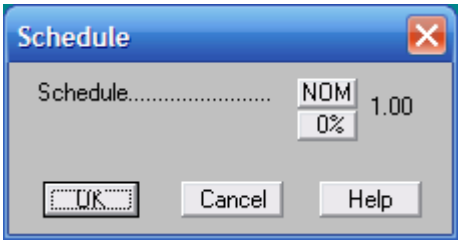
TDEV= [3.67 X 94.63 $^{(0.278+0.2x(1.20-1.01))}$] x 1

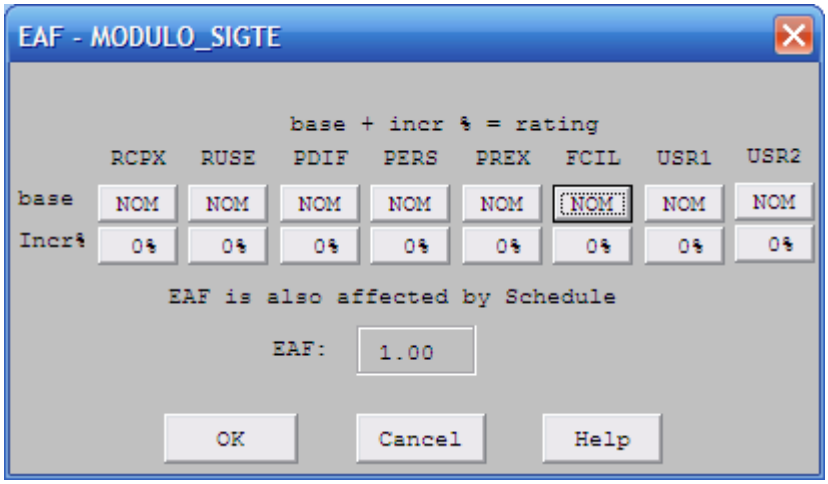
TDEV= [3.67 X 94.63 $^{0.316}$] x 1 = **15.5 Meses**

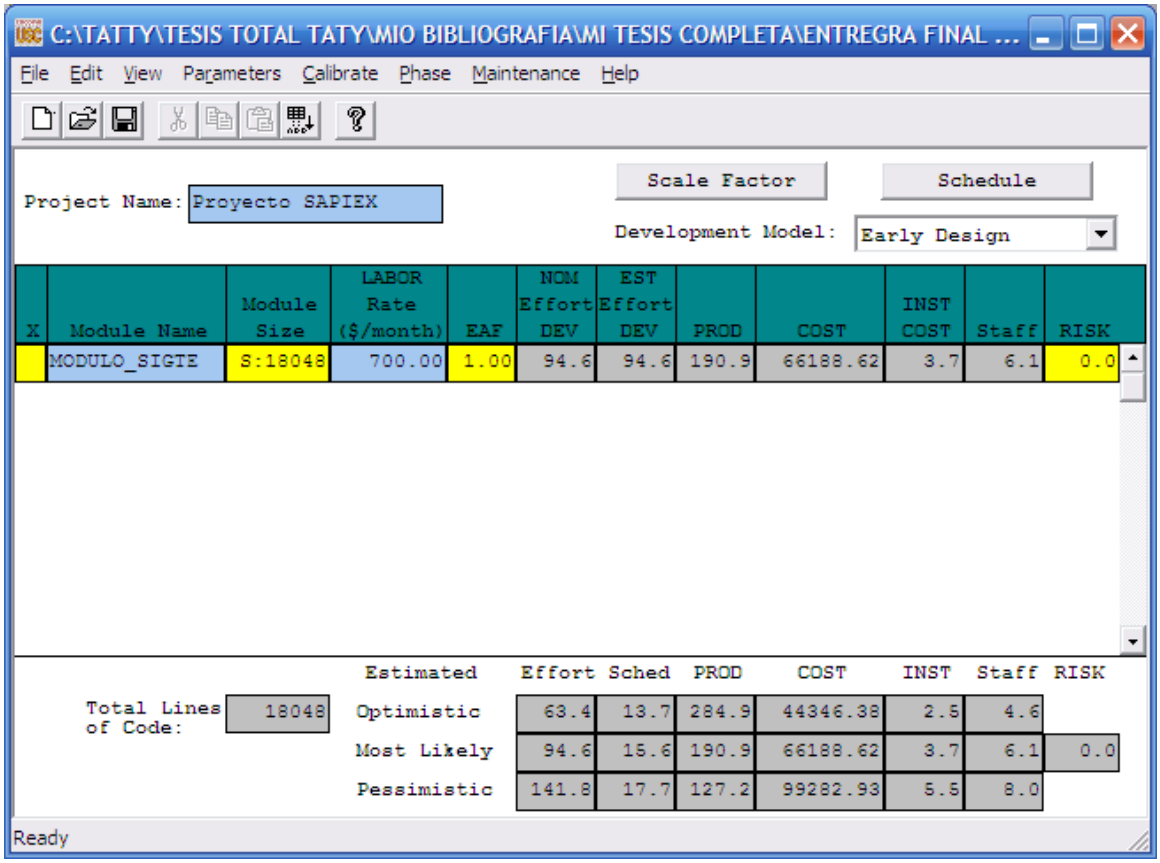
CÁLCULO MEDIANTE HERRAMIENTA COCOMO



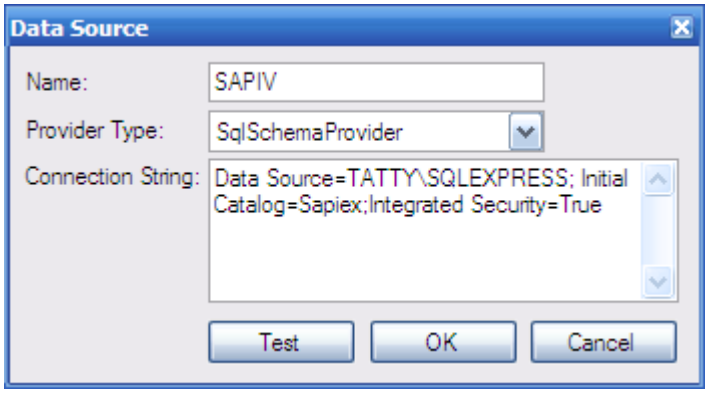

Scale Factors
Precedentedness ..... NOM 3.72
Development Flexibility ..... NOM 3.04
Architecture / risk resolution NOM 4.24
Team cohesion ..... NOM 3.29
Process maturity ..... NOM 4.68
OK
Cancel
Help

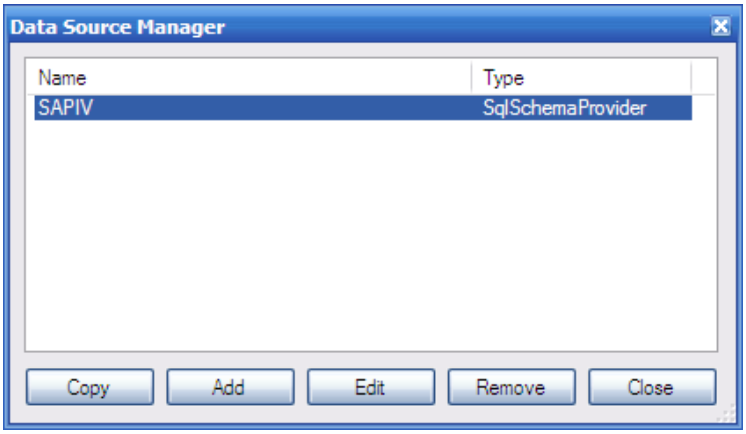

Schedule
Schedule..... NOM 0% 1.00
OK
Cancel
Help

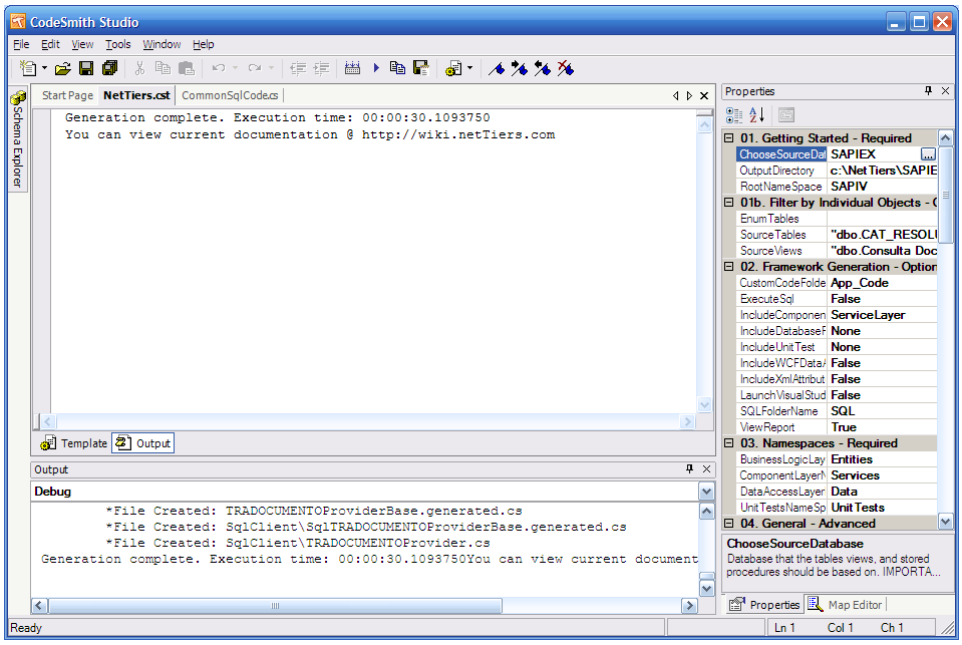

EAF - MODULO_SIGTE
base + incr % = rating
RCPX RUSE PDIF PERS PREX FCIL USR1 USR2
base NOM NOM NOM NOM NOM NOM NOM NOM
Incr% 0% 0% 0% 0% 0% 0% 0% 0%
EAF is also affected by Schedule
EAF: 1.00
OK
Cancel
Help



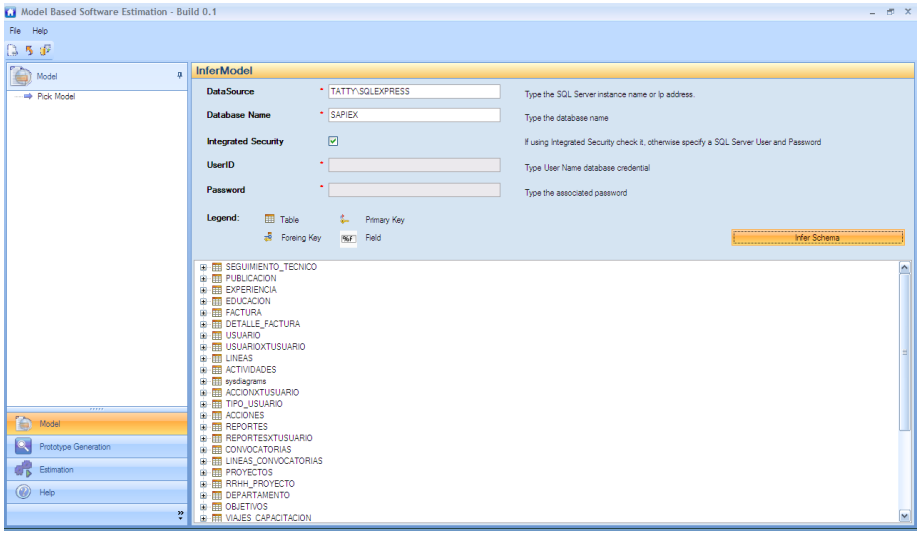
METODO PARA CALCULAR LA ARQUITECTURA DEL SISTEMA – CODESMITH STUDIO

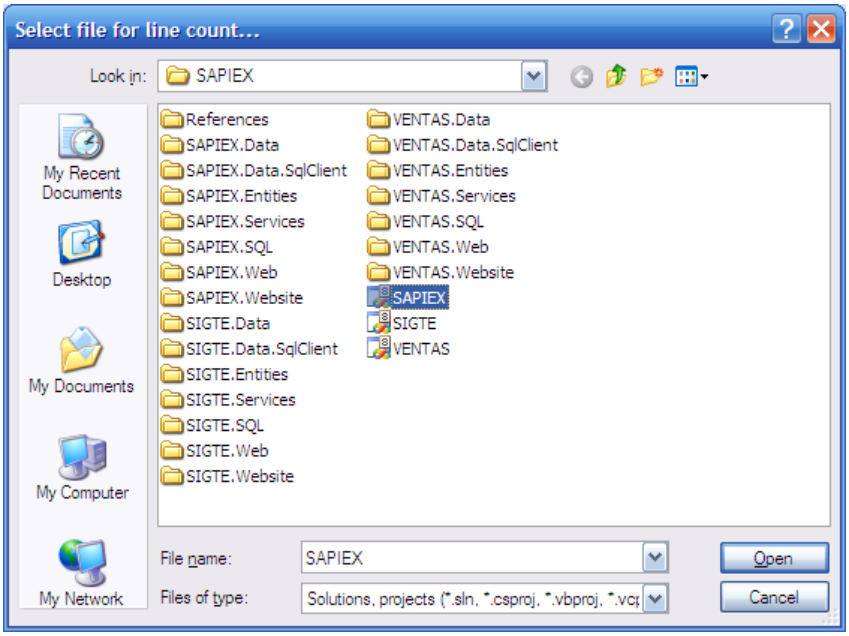


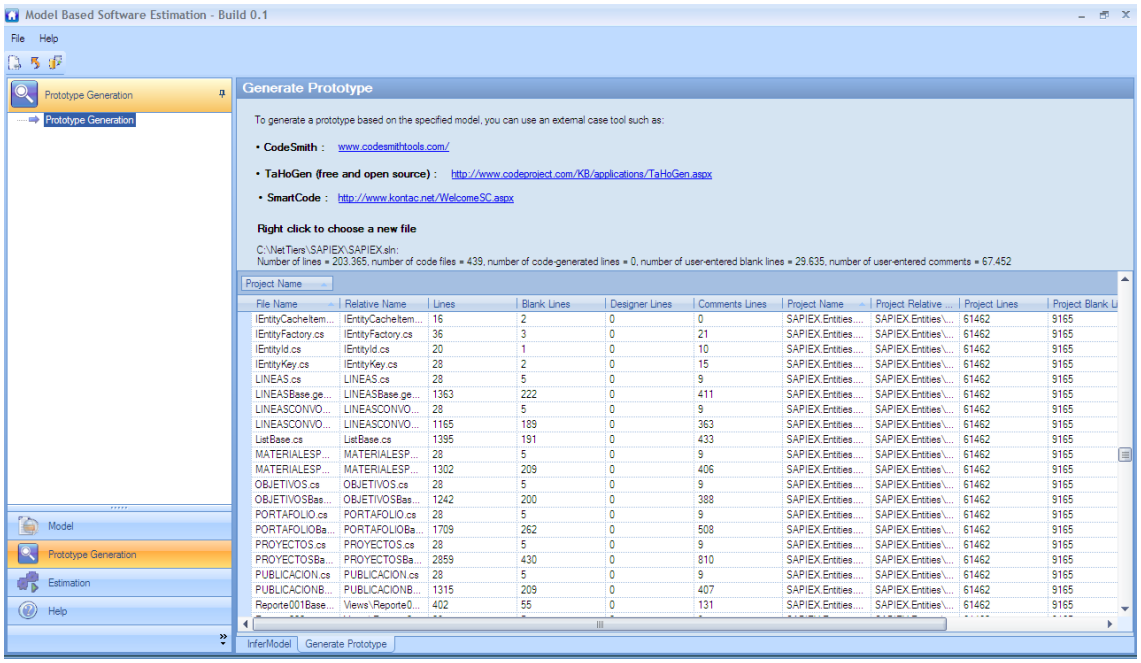




USO DE LA HERRAMIENTA MPE







TLOC= 105839

ALOC= TLOC-BLOC

ALOC= 105839-18048= 87791 SLOC

CALCULO DEL ESFUERZO DE LA ARQUITECTURA DEL SISTEMA

CÁLCULO DE ESFUERZO NOMINAL

$$PM_{nominal} = A \times (Size)^{B}$$

AJUSTE DE FACTORES DE ESCALA (B)

FÓRMULA:

$$B = 1.01 + 0.01 \times \sum_{j=1}^{5} Wj$$

VARIABLE	DESCRIPCION	PODERACIÓN	VALOR
PREC	El sistema es familiar.	Nominal	3.72
FLEX	La flexibilidad de desarrollo es moderada.	Nominal	3.04
RESL	La arquitectura es normal y los riesgos generalmente se mitigan.	Nominal	4.24
TEAM	La interacción del equipo es normal.	Nominal	3.29
PMAT	La madurez del proceso software es normal.	Nominal	4.68
		TOTAL	**1.20**

$PM_{nominal} = 2.94 \times (87.791)^{1.20}$

$PM_{nominal} = 631.6$ Meses-Persona

CÀLCULO DE ESFUERZO AJUSTADO

$$PM_{ajustado} = PM_{nominal} \times \Pi(ME_i)$$

AJUSTE DE DRIVERS DE COSTO

FÓRMULA:

$$\Pi(ME_i)$$

MULTIPLICADOR	DESCRIPCION	PONDERACIÓN	VALOR
PERS	Se tiene analistas y programadores con gran eficiencia y buen trabajo en equipo. Dedicación full-time.	Nominal	1
RCPX	La exigencia de confiabilidad, documentación y volumen de datos son moderadas, el producto no tiene mucha	Nominal	1

	complejidad.		
RUSE	Se pretende reutilizar nada.	Nominal	1
PDIF	Existen restricciones al tiempo de respuesta. No se especifica el hardware a utilizar, ni la plataforma de soporte.	Nominal	1
PREX	Todos los analistas y programadores tienen una moderada experiencia de desarrollo de la aplicación.	Nominal	1
SCED	Se requiere terminar el proyecto en el tiempo estimado.	Nominal	1
FCIL	Se requiere herramienta CASE para desarrollo Web. La comunicación de datos debe ser distribuido con conexión a Internet.	Nominal	1
		TOTAL	**1**

$PM_{ajustado}$= 631.6 x 1

$PM_{ajustado}$= 631.6 Meses-persona

CÁLCULO DEL TIEMPO DE DESARROLLO

$$TDEV = [3.67 \times PM^{(0.28+0.2\times(B-1.01))}] \times \frac{SCED\%}{100}$$

TDEV= [3.67 X 631.6 $^{(0.278+0.2x(1.20-1.01))}$] x 1

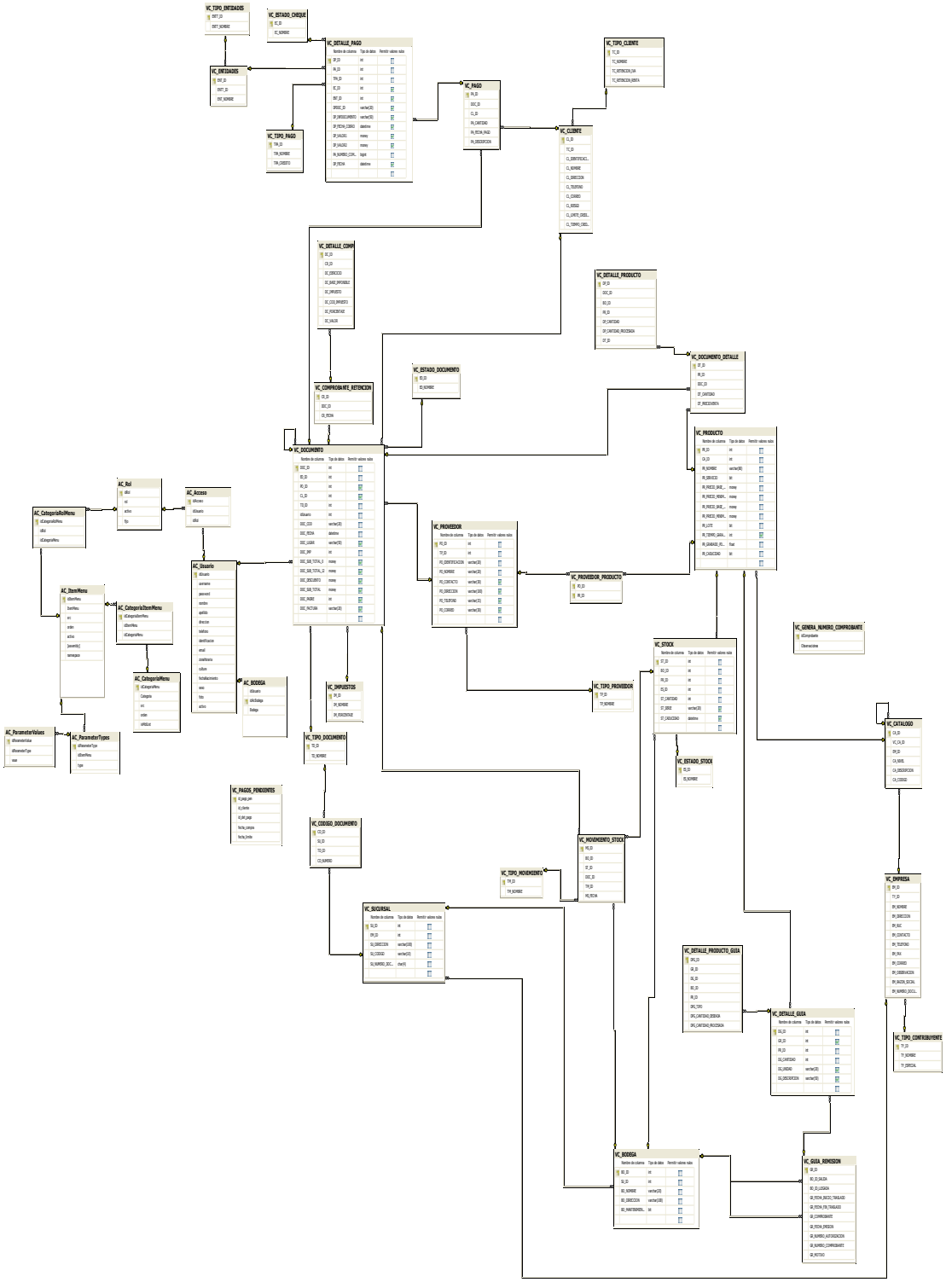
TDEV= [3.67 X 631.6 $^{0.316}$] x 1 **= 28.15 Meses**

ANEXO F
VENTAS: Sistema de gestión de Ventas

NUEVA PROPUESTA CÁLCULO DE PUNTOS DE FUNCIÓN

METODO PARA CALCULAR LA FUNCIONALIDAD DEL SISTEMA

MODELO ENTIDAD RELACION SISTEMA FACTURACIÓN Y VENTAS



ANÁLISIS DE LOS ELEMENTOS DE FUNCIÓN DEL PROYECTO FACTURACION Y VENTAS

Elemento de Función	Funciones	Tablas Afectadas FTRs	Tipos de Elementos de Datos DETs	Complejidad ILFs
Entrada Externa	Adicionar_VcTipoEntidades	1	3	Baja
Entrada Externa	Modificar_ VcTipoEntidades	1	3	Baja
Entrada Externa	Eliminar_ VcTipoEntidades	1	3	Baja
Consulta Externa	Consultar_ VcTipoEntidades	1	3	Baja
Entrada Externa	Adicionar_VcEntidades	2	4	Baja
Entrada Externa	Modificar_ VcEntidades	2	4	Baja
Entrada Externa	Eliminar_ VcEntidades	2	4	Baja
Consulta Externa	Consultar_VcEntidades	2	4	Baja
Entrada Externa	Adicionar_VcEstadoCheques	1	3	Baja
Entrada Externa	Modificar_ VcEstadoCheques	1	3	Baja
Entrada Externa	Eliminar_ VcEstadoCheques	1	3	Baja
Consulta Externa	Consultar_ VcEstadoCheques	1	3	Baja
Entrada Externa	Adicionar_VcTipoPago	1	4	Baja
Entrada Externa	Modificar_ VcTipoPago	1	4	Baja
Entrada Externa	Eliminar_ VcTipoPago	1	4	Baja
Consulta Externa	Consultar_ VcTipoPago	1	4	Baja
Entrada Externa	Adicionar_VcDetallePago	5	13	Alta
Entrada Externa	Modificar_ VcDetallePago	5	13	Alta
Entrada Externa	Eliminar_ VcDetallePago	5	13	Alta
Consulta Externa	Consultar_ VcDetallePago	5	13	Alta
Entrada Externa	Adicionar_VcPago	3	7	Alta

Entrada Externa	Modificar_ VcPago	3	7	Alta
Entrada Externa	Eliminar_ VcPago	3	7	Alta
Consulta Externa	Consultar_ VcPago	3	7	Media
Entrada Externa	Adicionar_VcCliente	2	11	Media
Entrada Externa	Modificar_ VcCliente	2	11	Media
Entrada Externa	Eliminar_ VcCliente	2	11	Media
Consulta Externa	Consultar_ VcCliente	2	11	Media
Entrada Externa	Adicionar_VcTipoCliente	1	5	Baja
Entrada Externa	Modificar_ VcTipoCliente	1	5	Baja
Entrada Externa	Eliminar_ VcTipoCliente	1	5	Baja
Consulta Externa	Consultar_ VcTipoCliente	1	5	Baja
Entrada Externa	Adicionar_VcDetalleProducto	2	8	Media
Entrada Externa	Modificar_ VcDetalleProducto	2	8	Media
Entrada Externa	Eliminar_ VcDetalleProducto	2	8	Media
Consulta Externa	Consultar_ VcDetalleProducto	2	8	Media
Entrada Externa	Adicionar_VcDocumentoDetalle	3	6	Alta
Entrada Externa	Modificar_ VcDocumentoDetalle	3	6	Alta
Entrada Externa	Eliminar_ VcDocumentoDetalle	3	6	Alta
Consulta Externa	Consultar_ VcDocumentoDetalle	3	6	Media
Entrada Externa	Adicionar_VcProducto	2	13	Media
Entrada Externa	Modificar_ VcProducto	2	13	Media
Entrada Externa	Eliminar_ VcProducto	2	13	Media
Consulta Externa	Consultar_ VcProducto	2	13	Media
Entrada Externa	Adicionar_ VcProveedorProducto	3	3	Media

Entrada Externa	Modificar_ VcProveedorProducto	3	3	Media
Entrada Externa	Eliminar_ VcProveedorProducto	3	3	Media
Consulta Externa	Consultar_ VcProveedorProducto	3	3	Baja
Entrada Externa	Adicionar_VcProveedor	2	9	Media
Entrada Externa	Modificar_ VcProveedor	2	9	Media
Entrada Externa	Eliminar_ VcProveedor	2	9	Media
Consulta Externa	Consultar_ VcProveedor	2	9	Media
Entrada Externa	Adicionar_VcTipoProveedor	1	3	Baja
Entrada Externa	Modificar_ VcTipoProveedor	1	3	Baja
Entrada Externa	Eliminar_ VcTipoProveedor	1	3	Baja
Consulta Externa	Consultar_ VcTipoProveedor	1	3	Baja
Entrada Externa	Adicionar_VcStock	4	8	Alta
Entrada Externa	Modificar_ VcStock	4	8	Alta
Entrada Externa	Eliminar_ VcStock	4	8	Alta
Consulta Externa	Consultar_ VcStock	4	8	Alta
Entrada Externa	Adicionar_VcMovimientoStock	5	7	Alta
Entrada Externa	Modificar_ VcMovimientoStock	5	7	Alta
Entrada Externa	Eliminar_ VcMovimientoStock	5	7	Alta
Consulta Externa	Consultar_ VcMovimientoStock	5	7	Alta
Entrada Externa	Adicionar_VcTipoMovimiento	1	3	Baja
Entrada Externa	Modificar_ VcTipoMovimiento	1	3	Baja
Entrada Externa	Eliminar_ VcTipoMovimiento	1	3	Baja
Consulta Externa	Consultar_ VcTipoMovimiento	1	3	Baja
Entrada Externa	Registrar_VcEstadoStock	1	3	Baja

Entrada Externa	Modificar_ VcEstadoStock	1	3	Baja
Entrada Externa	Eliminar_ VcEstadoStock	1	3	Baja
Consulta Externa	Consultar_ VcEstadoStock	1	3	Baja
Entrada Externa	Adicionar_VcGeneraNumeroComprobante	1	3	Baja
Entrada Externa	Modificar_VcGeneraNumeroComprobante	1	3	Baja
Entrada Externa	Eliminar_ VcGeneraNumeroComprobante	1	3	Baja
Consulta Externa	Consultar_VcGeneraNumeroComprobante	1	3	Baja
Entrada Externa	Adicionar_VcCatalogo	3	7	Alta
Entrada Externa	Modificar_ VcCatalogo	3	7	Alta
Entrada Externa	Eliminar_ VcCatalogo	3	7	Alta
Consulta Externa	Consultar_ VcCatalogo	3	7	Media
Entrada Externa	Adicionar_VcEmpresa	2	13	Media
Entrada Externa	Modificar_ VcEmpresa	2	13	Media
Entrada Externa	Eliminar_ VcEmpresa	2	13	Media
Consulta Externa	Consultar_ VcEmpresa	2	13	Media
Entrada Externa	Adicionar_VcTipoContribuyente	1	4	Baja
Entrada Externa	Modificar_ VcTipoContribuyente	1	4	Baja
Entrada Externa	Eliminar_ VcTipoContribuyente	1	4	Baja
Consulta Externa	Consultar_ VcTipoContribuyente	1	4	Baja
Entrada Externa	Adicionar_VcGuiaRemision	3	11	Alta
Entrada Externa	Modificar_ VcGuiaRemision	3	11	Alta
Entrada Externa	Eliminar_ VcGuiaRemision	3	11	Alta
Consulta Externa	Consultar_ VcGuiaRemision	3	11	Media
Entrada Externa	Adicionar_VcDetalleGuia	3	7	Alta

Entrada Externa	Modificar_ VcDetalleGuia	3	7	Alta
Entrada Externa	Eliminar_ VcDetalleGuia	3	7	Alta
Consulta Externa	Consultar_ VcDetalleGuia	3	7	Media
Entrada Externa	Adicionar_VcDetalleProductoGuia	2	9	Media
Entrada Externa	Modificar_ VcDetalleProductoGuia	2	9	Media
Entrada Externa	Eliminar_ VcDetalleProductoGuia	2	9	Media
Consulta Externa	Consultar_ VcDetalleProductoGuia	2	9	Media
Entrada Externa	Adicionar_VcBodega	2	6	Media
Entrada Externa	Modificar_ VcBodega	2	6	Media
Entrada Externa	Eliminar_ VcBodega	2	6	Media
Consulta Externa	Consultar_ VcBodega	2	6	Media
Entrada Externa	Adicionar_VcSucursal	2	6	Media
Entrada Externa	Modificar_ VcSucursal	2	6	Media
Entrada Externa	Eliminar_ VcSucursal	2	6	Media
Consulta Externa	Consultar_ VcSucursal	2	6	Media
Entrada Externa	Adicionar_VcCodigoDocumento	3	5	Alta
Entrada Externa	Modificar_ VcCodigoDocumento	3	5	Alta
Entrada Externa	Eliminar_ VcCodigoDocumento	3	5	Alta
Consulta Externa	Consultar_ VcCodigoDocumento	3	5	Baja
Entrada Externa	Adicionar_VcTipoDocumento	1	3	Baja
Entrada Externa	Modificar_ VcTipoDocumento	1	3	Baja
Entrada Externa	Eliminar_ VcTipoDocumento	1	3	Baja
Consulta Externa	Consultar_ VcTipoDocumento	1	3	Baja
Entrada Externa	Adicionar_VcImpuestos	1	4	Baja

Entrada Externa	Modificar_ VcImpuestos	1	4	Baja
Entrada Externa	Eliminar_ VcImpuestos	1	4	Baja
Consulta Externa	Consultar_ VcImpuestos	1	4	Baja
Entrada Externa	Adicionar_VcDocumento	8	17	Alta
Entrada Externa	Modificar_ VcDocumento	8	17	Alta
Entrada Externa	Eliminar_ VcDocumento	8	17	Alta
Consulta Externa	Consultar_ VcDocumento	8	17	Alta
Entrada Externa	Adicionar_VcEstadoDocumento	1	3	Baja
Entrada Externa	Modificar_ VcEstadoDocumento	1	3	Baja
Entrada Externa	Eliminar_ VcEstadoDocumento	1	3	Baja
Consulta Externa	Consultar_ VcEstadoDocumento	1	3	Baja
Entrada Externa	Adicionar_VcComprobanteRetencion	2	4	Baja
Entrada Externa	Modificar_ VcComprobanteRetencion	2	4	Baja
Entrada Externa	Eliminar_ VcComprobanteRetencion	2	4	Baja
Consulta Externa	Consultar_ VcComprobanteRetencion	2	4	Baja
Entrada Externa	Adicionar_VcDetalleCompra	2	9	Media
Entrada Externa	Modificar_ VcDetalleCompra	2	9	Media
Entrada Externa	Eliminar_ VcDetalleCompra	2	9	Media
Consulta Externa	Consultar_ VcDetalleCompra	2	9	Media
Entrada Externa	Adicionar_VcPagosPendientes	1	6	Baja
Entrada Externa	Modificar_ VcPagosPendientes	1	6	Baja
Entrada Externa	Eliminar_ VcPagosPendientes	1	6	Baja
Consulta Externa	Consultar_ VcPagosPendientes	1	6	Baja
Entrada Externa	Adicionar_AcBodega	2	4	Baja

Entrada Externa	Modificar_ AcBodega	2	4	Baja
Entrada Externa	Eliminar_ AcBodega	2	4	Baja
Consulta Externa	Consultar_ AcBodega	2	4	Baja
Entrada Externa	Adicionar_AcUsuario	1	16	Media
Entrada Externa	Modificar_ AcUsuario	1	16	Media
Entrada Externa	Eliminar_ AcUsuario	1	16	Media
Consulta Externa	Consultar_ AcUsuario	1	16	Baja
Entrada Externa	Adicionar_AcAcceso	3	4	Media
Entrada Externa	Modificar_ AcAcceso	3	4	Media
Entrada Externa	Eliminar_ AcAcceso	3	4	Media
Consulta Externa	Consultar_ AcAcceso	3	4	Baja
Entrada Externa	Adicionar_AcRol	1	5	Baja
Entrada Externa	Modificar_ AcRol	1	5	Baja
Entrada Externa	Eliminar_ AcRol	1	5	Baja
Consulta Externa	Consultar_ AcRol	1	5	Baja
Entrada Externa	Adicionar_AcCategoriaRolMenu	3	4	Media
Entrada Externa	Modificar_ AcCategoriaRolMenu	3	4	Media
Entrada Externa	Eliminar_ AcCategoriaRolMenu	3	4	Media
Consulta Externa	Consultar_ AcCategoriaRolMenu	3	4	Baja
Entrada Externa	Adicionar_ AcItemMenu	1	8	Baja
Entrada Externa	Modificar_ AcItemMenu	1	8	Baja
Entrada Externa	Eliminar_ AcItemMenu	1	8	Baja
Consulta Externa	Consultar_ AcItemMenu	1	8	Baja
Entrada Externa	Adicionar_AcCategoriaItemMenu	3	4	Media

Entrada Externa	Modificar_ AcCategoriaItemMenu	3	4	Media
Entrada Externa	Eliminar_ AcCategoriaItemMenu	3	4	Media
Consulta Externa	Consultar_ AcCategoriaItemMenu	3	4	Baja
Entrada Externa	Adicionar_AcCategoriaMenu	1	6	Baja
Entrada Externa	Modificar_ AcCategoriaMenu	1	6	Baja
Entrada Externa	Eliminar_ AcCategoriaMenu	1	6	Baja
Consulta Externa	Consultar_ AcCategoriaMenu	1	6	Baja
Entrada Externa	Adicionar_AcParameterType	2	4	Baja
Entrada Externa	Modificar_ AcParameterType	2	4	Baja
Entrada Externa	Eliminar_ AcParameterType	2	4	Baja
Consulta Externa	Consultar_ AcParameterType	2	4	Baja
Entrada Externa	Adicionar_AcParameterValues	2	4	Baja
Entrada Externa	Modificar_ AcParameterValues	2	4	Baja
Entrada Externa	Eliminar_ AcParameterValues	2	4	Baja
Consulta Externa	Consultar_ AcParameterValues	2	4	Baja

Archivos Lógicos ILFs	**Registros Lógicos**	**Tipos de Elementos de Datos DETs**	**Complejidad ILFs**
VC_TIPO_ENTIDADES	1	2	Baja
VC_ENTIDADES	1	3	Baja
VC_ESTADO_CHEQUE	1	2	Baja
VC_TIPO_PAGO	1	3	Baja
VC_DETALLE_PAGO	1	12	Baja
VC_PAGO	1	6	Baja
VC_CLIENTE	1	10	Baja

VC_TIPO_CLIENTE	1	4	Baja
VC_DETALLE_PRODUCTO	1	7	Baja
VC_DOCUMENTO_DETALLE	1	5	Baja
VC_PRODUCTO	1	12	Baja
VC_STOCK	1	7	Baja
VC_ESTADO_STOCK	1	2	Baja
VC_GENERA_NUMERO_COMPROBANTE	1	2	Baja
VC_CATALOGO	1	6	Baja
VC_EMPRESA	1	12	Baja
VC_TIPO_CONTRIBUYENTE	1	3	Baja
VC_GUIA_REMISION	1	10	Baja
VC_DETALLE_GUIA	1	6	Baja
VC_DETALLE_PRODUCTO_GUIA	1	8	Baja
VC_BODEGA	1	5	Baja
VC_PROVEEDOR_PRODUCTO	1	2	Baja
VC_TIPO_PROVEEDOR	1	2	Baja
VC_MOVIMIENTO_STOCK	1	6	Baja
VC_TIPO_MOVIMIENTO	1	2	Baja
VC_PROVEEDOR	1	8	Baja
VC_ESTADO_DOCUMENTO	1	2	Baja
VC_DOCUMENTO	1	16	Baja
VC_COMPROBANTE_RETENCION	1	3	Baja
VC_DETALLE_COMPRA	1	8	Baja
VC_REPUESTOS	1	3	Baja
VC_TIPO_DOCUMENTO	1	2	Baja

VC_CODIGO_DOCUMENTO	1	4	Baja
VC_SUCURSAL	1	5	Baja
VC_PAGOS_PENDIENTES	1	5	Baja
AC_BODEGA	1	3	Baja
AC_USUARIO	1	15	Baja
AC_ACCESO	1	3	Baja
AC_ROL	1	4	Baja
AC_CATEGORIA_ITEM_MENU	1	3	Baja
AC_CATEGORIA_MENU	1	5	Baja
AC_ITEM_MENU	1	7	Baja
AC_CATEGORIA_ROL_MENU	1	3	Baja
AC_PARAMETER_TYPES	1	3	Baja
AC_PARAMETER_VALUES	1	3	Baja

ENTRADAS EXTERNAS

Archivos referenciados	**Elementos de Datos**		
	1-4	5-15	>15
0-1	Baja	Baja	Media
2	Baja	Media	Alta
3 ó más	Media	Alta	Alta(3)

CONSULTAS EXTERNAS

Archivos referenciados	**Elementos de Datos**		
	1-5	6-19	>19
0-1	Baja	Baja	Media
2-3	Baja	Media	Alta
>3	Media	Alta	Alta

ARCHIVOS LÓGICOS INTERNOS

Tipos de Registros	Elementos de Datos		
	1-19	20-50	>50
1	Baja	Baja	Media
2-5	Baja	Media	Alta
>5	Media	Alta	Alta

PUNTOS DE FUNCION SIN AJUSTAR (PFSA)

Punto de Función	Complejidad			
	Alta	**Media**	**Baja**	**Total**
Entradas Externas	6***30**	4***42**	3***63**	537
Salidas Externas	7	5	4	0
Consultas Externas	6***4**	4***14**	3***26**	158
Archivos Lógicos Internos	15	10	7***45**	315
Interfases Externas	10	7	5	0
			PFSA	**1010**

CARACTERISTICAS GENERALES DEL SISTEMA

Valor	Significado
0	Sin influencia, factor no presente
1	Influencia insignificante, muy baja
2	Influencia moderada o baja
3	Influencia media, normal
4	Influencia alta, significativa
5	Influencia muy alta, esencial

Características Generales del Sistema		Complejidad	Comentario
1	Comunicación de datos	3	Comunicación de datos moderada.
2	Procesamiento de datos distribuido	1	Sistema distribuido con conexión remota o directa a la red.
3	Rendimiento	2	Tiempos de respuesta que no superen los 10 segundos en redes extendidas y menores que 3 segundos sobre redes locales.
4	Uso de hardware existente	1	No se ha detallado.
5	Transacciones	3	Moderado manejo de transacciones.

6	Entrada de datos interactiva	4	Manejo de browser de Internet para la entrada de datos.
7	Eficiencia	4	Si existe eficiencia del sistema.
8	Actualizaciones on-line	4	Manejo de Internet para ejecutar transacciones.
9	Complejidad de procesamiento	1	Hay pocos cálculos en el sistema.
10	Reusabilidad	0	No se requiere que el código sea reutilizable
11	Facilidad de conversión e instalación	3	Los usuarios disponen de terminales con la configuración mínima requerida por los navegadores web.
12	Facilidad de operación	3	Operación normal.
13	Múltiples instalaciones	1	Se requiere escasas instalaciones del sistema en todos los equipos.
14	Facilidad de mantenimiento	3	Se requiere un costo moderado de mantenimiento.
Valor de Complejidad VAF		**33**	

CÁLCULOS DE PUNTOS DE FUNCION AJUSTADO:

PFSA= Punto de Función sin ajustar= 1010

PFA= PFSA *((VAF*0.01)+0.65)

VAF= Grado Total de Complejidad= 33

- PFA= 1010((33*0.01)+0.65)= 989.8

TRANSFORMACIÓN PF A SLOC

Size = 32 (VB) x 1010 PFSA = 32320 SLOC = **32.32 KSLOC**

CALCULO DEL ESFUERZO FUNCIONAL DEL SISTEMA

CÁLCULO DE ESFUERZO NOMINAL

$$\mathbf{PM_{nominal} = A \times (Size)^{B}}$$

AJUSTE DE FACTORES DE ESCALA (B)

FÓRMULA:

$$B = 1.01 + 0.01 \times \sum_{j=1}^{5} Wj$$

VARIABLE	DESCRIPCION	PODERACIÓN	VALOR
PREC	El sistema es familiar.	Nominal	3.72

FLEX	La flexibilidad de desarrollo es moderada.	Nominal	3.04
RESL	La arquitectura es normal y los riesgos generalmente se mitigan.	Nominal	4.24
TEAM	La interacción del equipo es normal.	Nominal	3.29
PMAT	La madurez del proceso software es normal.	Nominal	4.68
		TOTAL	**1.20**

$PM_{nominal} = 2.94 \times (32.32)^{1.20}$

$PM_{nominal} = 190.42$ Meses-Persona

CÀLCULO DE ESFUERZO AJUSTADO

$$PM_{ajustado} = PM_{nominal} \times \Pi(ME_i)$$

AJUSTE DE DRIVERS DE COSTO

FÓRMULA:

$$\Pi(ME_i)$$

MULTIPLICADOR	DESCRIPCION	PONDERACIÓN	VALOR
PERS	Se tiene analistas y programadores con gran eficiencia y buen trabajo en equipo. Dedicación full-time.	Nominal	1
RCPX	La exigencia de confiabilidad, documentación y volumen de datos son moderadas, el producto no tiene mucha complejidad.	Nominal	1
RUSE	Se pretende reutilizar nada.	Nominal	1
PDIF	Existen restricciones al tiempo de respuesta. No se especifica el hardware a utilizar, ni la plataforma de soporte.	Nominal	1
PREX	Todos los analistas y programadores tienen una moderada experiencia de desarrollo de la aplicación.	Nominal	1
SCED	Se requiere terminar el proyecto en el	Nominal	1

	tiempo estimado.		
FCIL	Se requiere herramienta CASE para desarrollo Web. La comunicación de datos debe ser distribuido con conexión a Internet.	Nominal	1
		TOTAL	**1**

$PM_{ajustado}$= 190.42 x 1

$PM_{ajustado}$= 190.42 Meses-persona

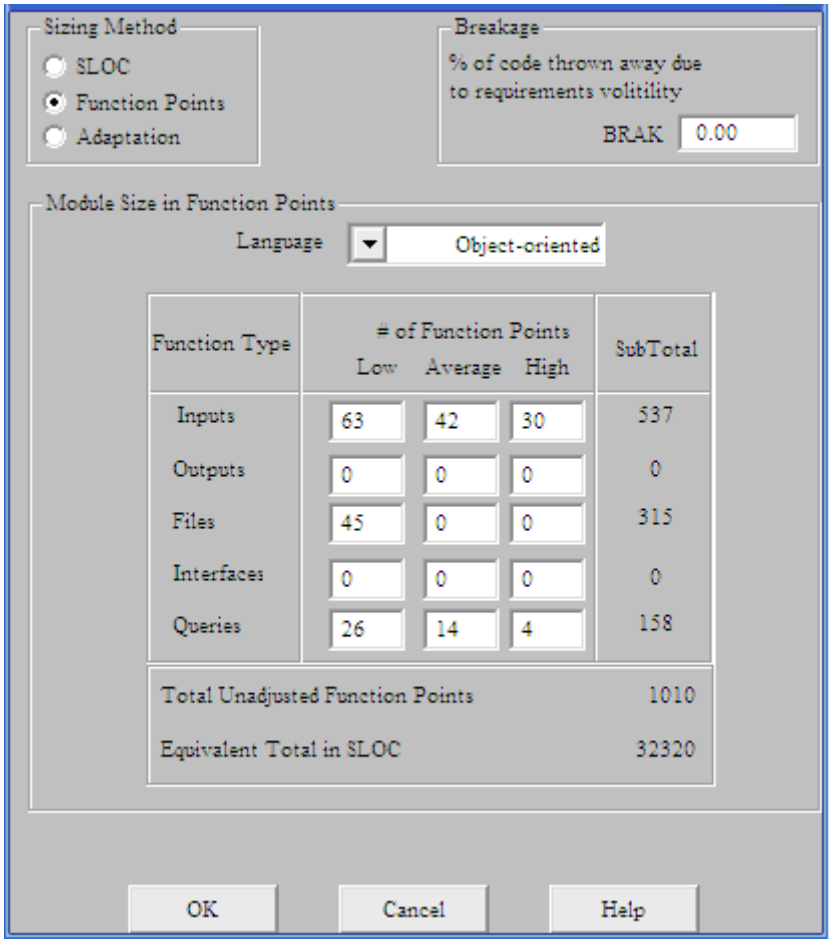
CÁLCULO DEL TIEMPO DE DESARROLLO

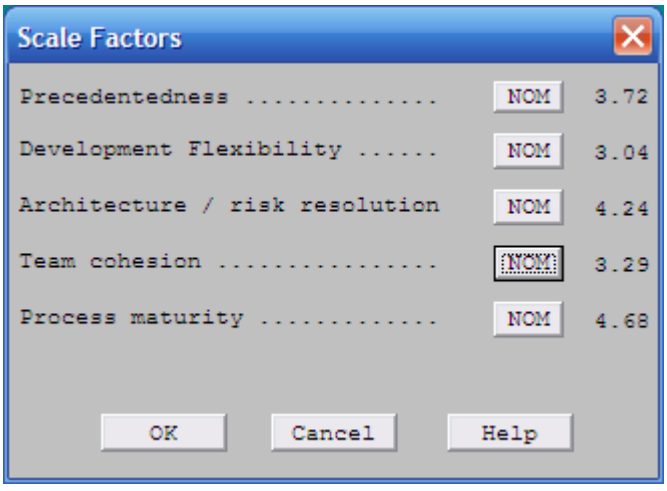
$$TDEV = [3.67 \times PM^{(0.28+0.2\times(B-1.01))}] \times \frac{SCED\%}{100}$$

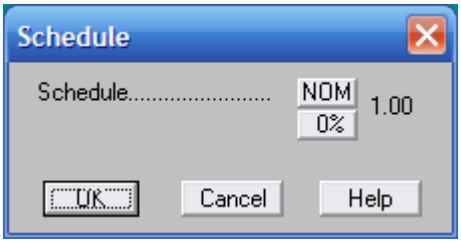
TDEV= [3.67 X 94.63 $^{(0.278+0.2x(1.20-1.01))}$] x 1

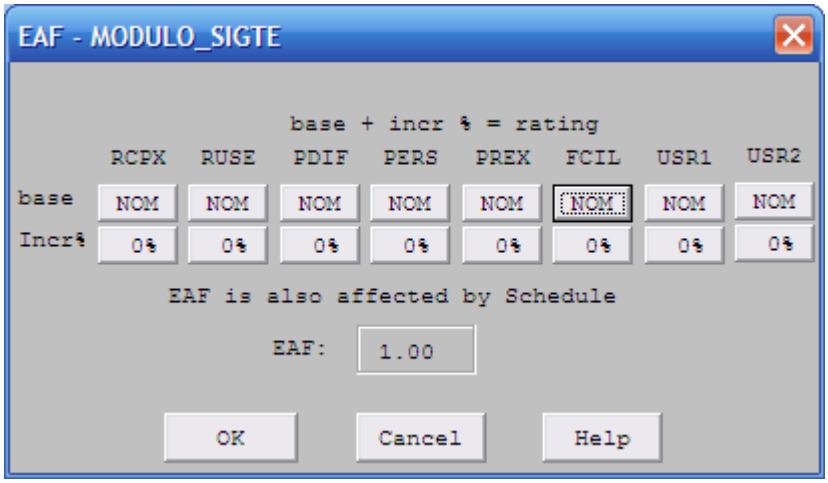
TDEV= [3.67 X 190.42 $^{0.316}$] x 1 **= 19.3 Meses**

CÁLCULO MEDIANTE HERRAMIENTA COCOMO

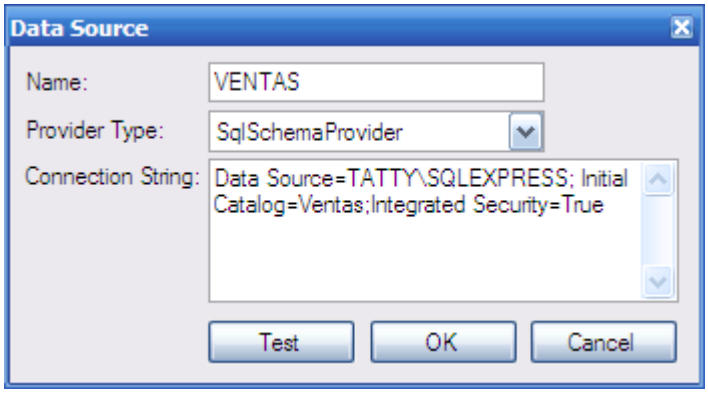

Sizing Method
SLOC
Function Points
Adaptation
Breakage
% of code thrown away due to requirements volitility
BRAK 0.00
Module Size in Function Points
Language Object-oriented
Function Type
# of Function Points
Low Average High
SubTotal
Inputs 63 42 30 537
Outputs 0 0 0 0
Files 45 0 0 315
Interfaces 0 0 0 0
Queries 26 14 4 158
Total Unadjusted Function Points 1010
Equivalent Total in SLOC 32320
OK
Cancel
Help

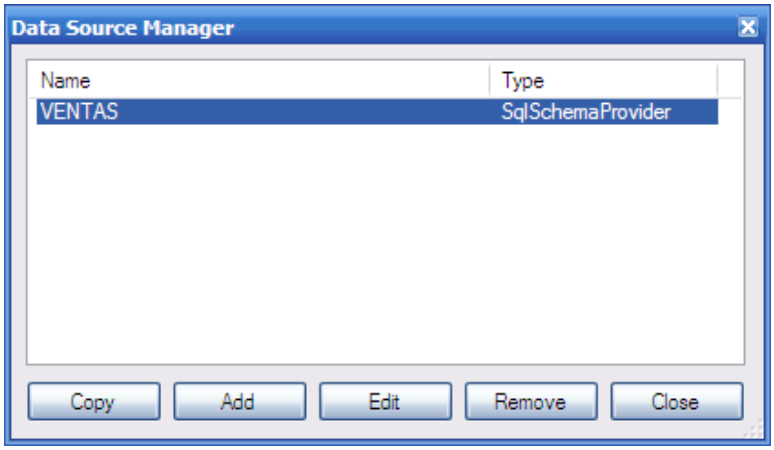

Scale Factors
Precedentedness ..... NOM 3.72
Development Flexibility ..... NOM 3.04
Architecture / risk resolution NOM 4.24
Team cohesion ..... NOM 3.29
Process maturity ..... NOM 4.68
OK
Cancel
Help

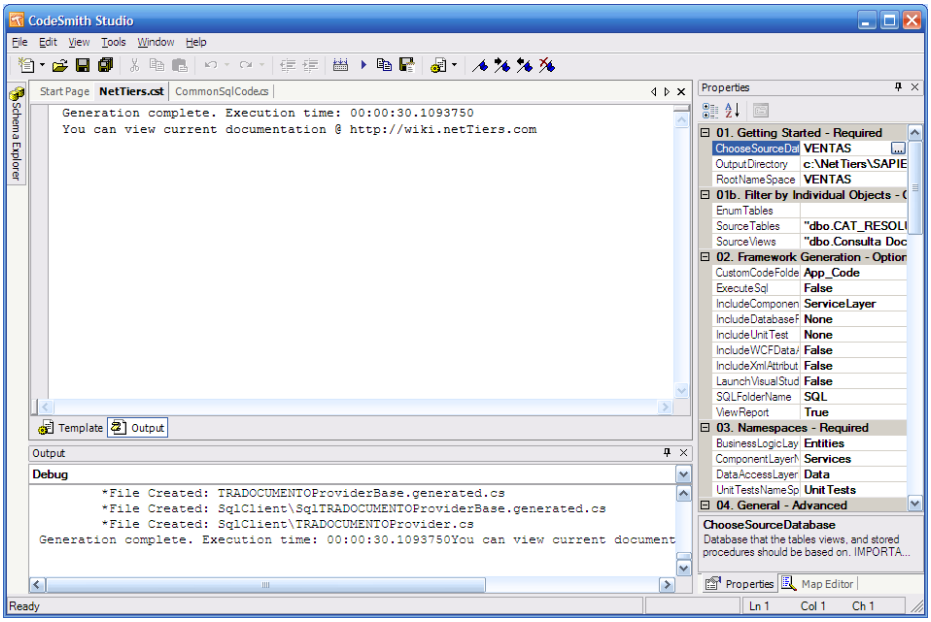

Schedule
Schedule..... NOM 0% 1.00
OK
Cancel
Help



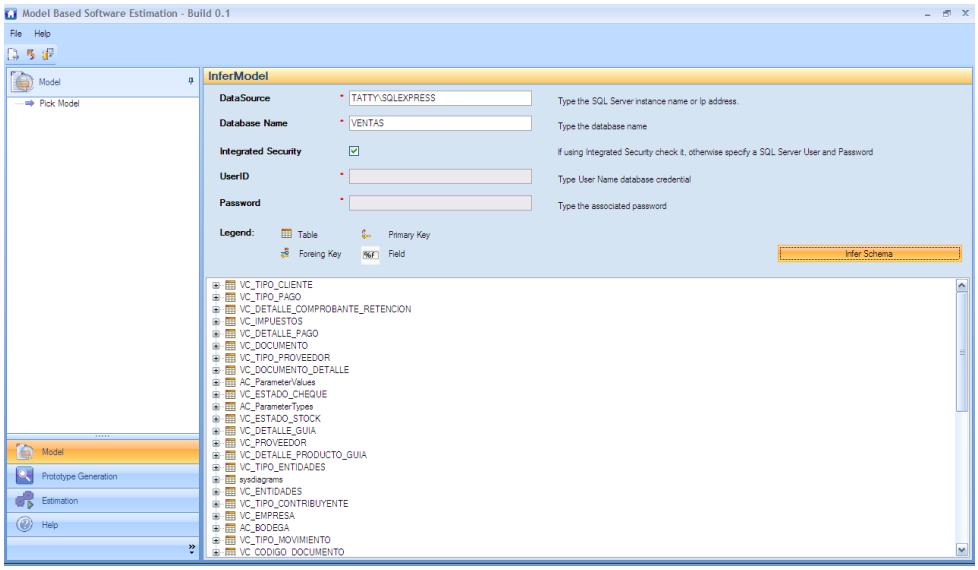
METODO PARA CALCULAR LA ARQUITECTURA DEL SISTEMA – CODESMITH STUDIO

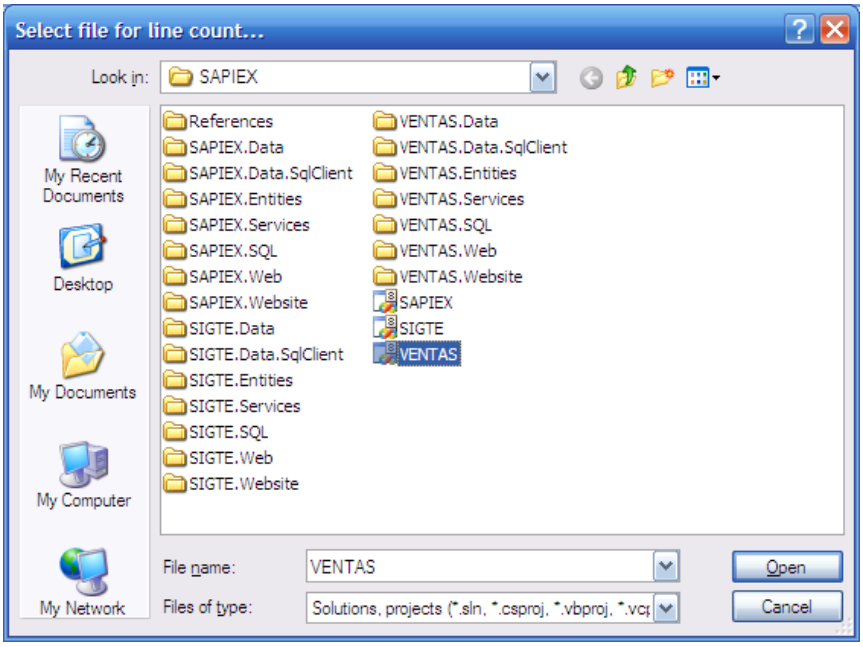


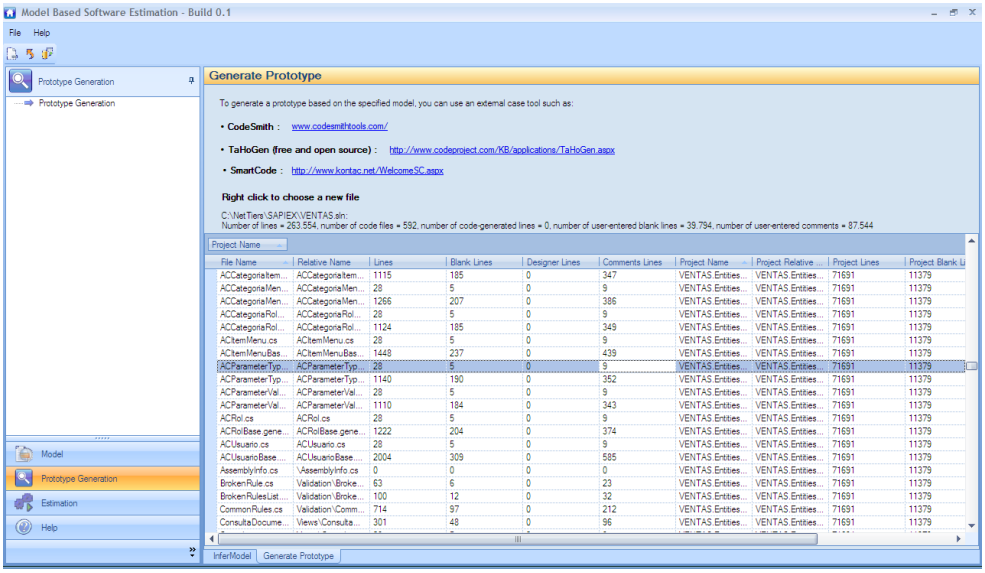




USO DE LA HERRAMIENTA MPE







TLOC= 135624

ALOC= TLOC-BLOC

ALOC= 135624-32320= 103304 SLOC

CALCULO DEL ESFUERZO DE LA ARQUITECTURA DEL SISTEMA

CÁLCULO DE ESFUERZO NOMINAL

$$PM_{nominal} = A \times (Size)^{B}$$

AJUSTE DE FACTORES DE ESCALA (B)

FÓRMULA:

$$B = 1.01 + 0.01 \times \sum_{j=1}^{5} Wj$$

VARIABLE	DESCRIPCION	PODERACIÓN	VALOR
PREC	El sistema es familiar.	Nominal	3.72
FLEX	La flexibilidad de desarrollo es moderada.	Nominal	3.04
RESL	La arquitectura es normal y los riesgos generalmente se mitigan.	Nominal	4.24
TEAM	La interacción del equipo es normal.	Nominal	3.29
PMAT	La madurez del proceso software es normal.	Nominal	4.68
		TOTAL	**1.20**

$PM_{nominal} = 2.94 \times (103.304)^{1.20}$

$PM_{nominal} = 767.87$ Meses-Persona

CÀLCULO DE ESFUERZO AJUSTADO

$$PM_{ajustado} = PM_{nominal} \times \Pi(ME_i)$$

AJUSTE DE DRIVERS DE COSTO

FÓRMULA:

$$\Pi(ME_i)$$

MULTIPLICADOR	DESCRIPCION	PONDERACIÓN	VALOR
PERS	Se tiene analistas y programadores con gran eficiencia y buen trabajo en equipo. Dedicación full-time.	Nominal	1
RCPX	La exigencia de confiabilidad, documentación y volumen de datos son moderadas, el producto no tiene mucha complejidad.	Nominal	1

RUSE	Se pretende reutilizar nada.	Nominal	1
PDIF	Existen restricciones al tiempo de respuesta. No se especifica el hardware a utilizar, ni la plataforma de soporte.	Nominal	1
PREX	Todos los analistas y programadores tienen una moderada experiencia de desarrollo de la aplicación.	Nominal	1
SCED	Se requiere terminar el proyecto en el tiempo estimado.	Nominal	1
FCIL	Se requiere herramienta CASE para desarrollo Web. La comunicación de datos debe ser distribuido con conexión a Internet.	Nominal	1
		TOTAL	**1**

$PM_{ajustado}$= 767.87 x 1

$PM_{ajustado}$= 767.87 Meses-persona

CÁLCULO DEL TIEMPO DE DESARROLLO

$$TDEV = [3.67 \times PM^{(0.28+0.2\times(B-1.01))}] \times \frac{SCED\%}{100}$$

TDEV= [3.67 X 767.87 $^{(0.278+0.2x(1.20-1.01))}$] x 1

TDEV= [3.67 X 767.87 $^{0.316}$] x 1 **= 29.95 Meses**

HOJA DE LEGALIZACION DE FIRMAS

ELABORADO POR

ROSA TATIANA VALENZUELA VARELA

COORDINADOR DE LA CARRERA

RAMIRO DELGADO

Lugar y fecha: Sangolquí, 26 de Septiembre del 2008