ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"MEDICIÓN DE LA PRODUCTIVIDAD DE PROYECTOS DE SOFTWARE DESARROLLADOS EN DOS EMPRESAS LOCALES."

INFORME DE PROYECTO DE GRADUACIÓN

Previo a la obtención del Título de:

INGENIERO EN COMPUTACIÓN ESPECIALIZACIÓN SISTEMAS MULTIMEDIA

Presentado por:

LOHANA MARIELLA LEMA MORETA

MANUEL ERNESTO OLVERA ALEJANDRO

Guayaquil – Ecuador

2010

AGRADECIMIENTO

A Dios,

a nuestros queridos y compresivos padres,

a nuestros abuelos y demás familiares por su apoyo,

a nuestra directora de tesis Ingeniera Mónica Villavicencio por su ardua labor como docente, directora de tesis y profesional; sus enseñanzas son invalorables,

a todos los docentes que aportaron con su conocimiento a lo largo de nuestra formación profesional.

DEDICATORIA

A nuestros padres:

Sr. Wilson Lema B. y Sra. Mariella Moreta M.

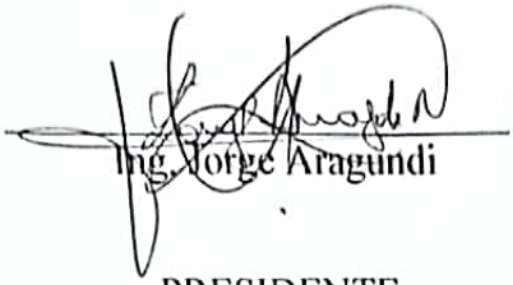
Sr. Manuel Olvera S y Sra. Mónica Alejandro,

y hermanos:

Wilson Lema Moreta y Penélope Lema Moreta

Joice Olvera Alejandro.

TRIBUNAL DE GRADO



Ing. Jorge Aragundi

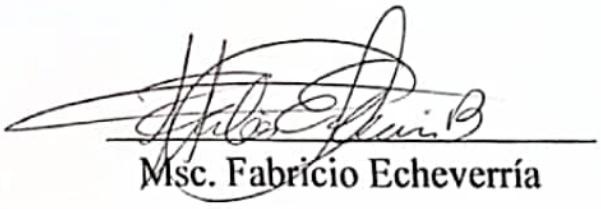
PRESIDENTE

Msc. Mónica Villavicencio Cabezas

DIRECTORA DE PROYECTO DE GRADUACIÓN

Ing. Verónica Macías

Miembro Principal



Msc. Fabricio Echeverría

Miembro Principal

DECLARACION EXPRESA

"La responsabilidad del contenido de esta Proyecto de Grado, me corresponde exclusivamente; y el patrimonio intelectual de la misma a la Escuela Superior Politécnica del Litoral".

Lohana Mariella Lema Moreta.

Manuel Ernesto Olvera Alejandro.

RESÚMEN

El problema a estudiar en este proyecto de graduación consiste medir la productividad y la eficiencia en ambientes de desarrollo de software, aprovechando el énfasis que las empresas ecuatorianas desean poner en aumentar su eficiencia, reducir costos y sobretodo estimar el tiempo y recursos que deberán asignar a sus proyectos. Este estudio se realizó en dos empresas desarrolladoras de software en Guayaquil, de las cuales extrajimos información de sus proyectos y de los ambientes en que estos se generaban, siendo éstos la base fundamental para medir la productividad en la generación de productos de software. Para lograr lo mencionado, realizamos una capacitación sobre un método de estimación como medida de productividad en un ambiente de desarrollo de Software, posteriormente definimos la productividad como una comparación entre el esfuerzo actual y el esfuerzo estimado por el modelo escogido, finalizando con un análisis de los resultados obtenidos, los cuales demostrarán la aplicabilidad del uso del modelo COCOMO II.

Este estudio analiza de forma real y metódica la productividad en las empresas participantes. La metodología consideró el análisis de los modelos de estimación de esfuerzo que fueron aplicados en 2 casos de estudio.

Este trabajo se sustenta en la tesis doctoral realizada por la PhD. Lotte de Rore en KULEUVEN -Katholieke Universiteit Leuven de Bélgica, quien emitió criterios

sobre los resultados obtenidos en Ecuador y obtuvo el cálculo de la productividad empleando los valores ajustados del modelo COCOMO II basado en 22 proyectos analizados de KBC Bank.

ÍNDICE GENERAL

ABREVIATURAS Y SIGLAS

COCOMO: Constructive Cost Model.

IFPUG: International Function Point User Group

FP: Function Points

SF: Scale Factors – Factores de Escala

LOC: Líneas de Código

KSLOC: Miles de Líneas de Código.

SEI: Instituto de Ingeniería de Software.

AT: Cantidad de código generado automáticamente.

DM: Porcentaje del Modelo que ha sido modificado.

CM: El porcentaje de código que ha sido modificado.

IM: Porcentaje de Esfuerzo necesario para integrar el código adaptado o reutilizado.

AAF: Cantidad de esfuerzo para modificar el software existente.

SU: Comprensibilidad del Software existente.

UNFM: Falta de Familiaridad del programados con el software existente.

AA: Esfuerzo necesario para decidir si un módulo es adecuado para la reutilización.

IM: Esfuerzo necesario para integrar el software necesario con el proyecto en conjunto.

EM: Effort Multipliers – Multiplicadores de Esfuerzo.

PM: Personas Mes

A y B: Factores constantes de COCOMO.

LTEX: Experiencia en el Lenguaje y Herramientas.

PLEX: Experiencia en la Plataforma.

DATA: Tamaño de la Base de Datos.

STOR: Restricción en el Almacenamiento Principal.

TIME: Restricción en el Tiempo de Ejecución.

RELY: Requerimiento de Fiabilidad de Software.

CPLX: Complejidad del Producto.

RUSE: Desarrollo para reusabilidad.

DOCU: Documentación en el Ciclo de Vida.

APEX: Experiencia en la aplicación.

TOOL: Uso de las Herramientas de Software.

3GL: Lenguajes de Tercera Generación.

ÍNDICE DE TABLAS

ÍNDICE DE FÓRMULAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La medición está presente en todas nuestras actividades diarias: cuando pensamos en los costos de la vida, cuando hablamos de comparar distancias entre dos puntos, e incluso cuando preparamos algún alimento; las medidas nos ayudan a entender nuestro ambiente para poder compararlo con otros (1).

Las medidas que tomamos a diario, nos ayudan a entregar información o para describir los atributos de un objeto determinado. Con el pasar del tiempo nos volvemos expertos en realizar mediciones, los que nos permite moldear conceptos y hacerlos más comprensibles y controlables e incluso especializarnos en algo en particular (1); aunque existen ocasiones que pensamos que los atributos son imposibles de medir, nos equivocamos en no intentar medirlo, pues no contribuye a nuestro intento de mejorar la compresión de los atributos y objetos.

Especialmente en el área de ingeniería de software, muchas de las cosas que en ella se ven involucradas, podrían considerarse no medibles (1). Por ejemplo, es difícil cuantificar a un buen software o un proyecto exitoso. Para evaluar la situación de los proyectos, productos o procesos, necesitamos mediciones en ingeniería de software. Como Fenton explica (1), estas mediciones nos ayudan a comprender lo que está pasando en los proyectos, a controlar lo que está sucediendo y sobretodo nos anima a mejorar los procesos y productos.

Una métrica muy importante dentro del área de ingeniería de software es la productividad, que es lo que mediremos en el presente proyecto.

Un modelo de estimación es lo que se utiliza como norma o punto de referencia para el cálculo de la productividad de los proyectos de software, su uso correcto y la preparación adecuada, dependiendo del ambiente en que se lo utilice, puede llegar a reducir considerablemente el grado de incertidumbre de su resultado. El modelo de estimación seleccionado para el presente proyecto de graduación es COCOMO II, cuya mayor fortaleza es su larga lista de manejadores de costo (factores de escala, y multiplicadores de esfuerzo). Con este modelo implementaremos un proceso completo en las empresas participantes y además incluiremos las decisiones, contratiempos y compensaciones que se realizarán para finalmente identificar las aéreas que se pueden mejorar para incrementar la productividad dentro del proceso de desarrollo de cada una de ellas.

CAPÍTULO 1

1. PANORAMA GENERAL DEL PROYECTO

1.1 Contexto y definición del problema.

El tema de la productividad de software ha generado interés por años.

Este interés puede ser situado en 4 áreas (2): 1) medición de la productividad, 2) predicción del esfuerzo, 3) productividad de una organización frente otras organizaciones y 4) exploración de los factores que afectan la productividad de software de una manera positiva o negativa, permitiendo tomar acciones correctivas. Sin embargo, el amplio interés en el dominio de la productividad de software, no ha permitido aún tener respuestas realmente claras para ninguna de las cuatro áreas especificadas anteriormente.

En este proyecto, analizaremos una de estas áreas, la primera, es decir nos direccionaremos al problema de cómo medir la productividad en el desarrollo de software y subsecuentemente, el problema de cómo mejorar el estudio de ésta dentro de un ambiente de desarrollo empresarial.

Este estudio se lleva a cabo en el contexto de 2 empresas desarrolladoras de software del mercado ecuatoriano. Las empresas participantes, nos permitirán tomar datos de 4 proyectos, 2 por empresa. Al finalizar el desarrollo del proyecto, estas empresas podrán tener una idea clara del estado actual en el ámbito de la productividad y además conocer qué

medidas correctivas se pueden poner en práctica para mejorar o mantener su nivel de productividad. Es importante recalcar, que es la primera vez que estas empresas se someten a éste tipo de estudio.

1.2 Medición y mejora de la productividad del desarrollo del software.

Si hablamos de la definición de Productividad podemos, por ejemplo, citar que se trata de la proporción de las salidas sobre las entradas, es decir la aproximación tradicional de la medición de la productividad del software mediante el cálculo de la proporción del tamaño sobre el esfuerzo.

Las salidas podrían ser medidas como el número de productos que se entregan, lo cual también es aplicable a la ingeniería de software, sin embargo debemos definir cómo interpretar “las entradas” y “las salidas” en este ámbito.

Una Entrada, es la cantidad de esfuerzo que nosotros gastamos hasta la entrega de un software. Medir la cantidad de las “salidas” no es algo que se pueda medir directamente, no existe una fórmula que indique el resultado de lo que exactamente se produce en un proyecto de software (3); es por esto que parámetros como: la cantidad de las líneas de código que se producen, la funcionalidad del producto cuando es entregado, los atributos de calidad que cumple el producto, entre otras son consideradas como medidas que tratan de cuantificar el tamaño de un buen producto de

Software, lo que implica que la productividad del software se vea expresada como el tamaño sobre el esfuerzo.

El esfuerzo puede ser medido como las horas trabajadas en el desarrollo del software.

El tamaño se vuelve algo más complicado de medir, existen muchos métodos de medir el tamaño del software, entre éstos podemos nombrar: Líneas de Código, IFPUG Puntos de Función (4) (5) o Cosmic Full Puntos de Función (6) (7). Históricamente el método más usado es el de líneas de software. Es importante recalcar que Function Points y Cosmic Full no son modelos de estimación como tal, son modelos de medición del tamaño de software. El tamaño que se puede predecir con éstos modelos puede ser usado para hacer una predicción del esfuerzo. Desafortunadamente, ninguno de los métodos nombrados son óptimos en el contexto de la medición de la productividad.

Usar el método de Líneas de Código, puede traer implícito muchas paradojas (3), por ejemplo: contar las líneas de código dependiendo del lenguaje de programación, contar las líneas de código depende del estilo de programación, entre otras.

El método IFPUG Puntos de Función (5) (4) es muy orientado a una plataforma específica y falla al capturar resultados de mejora de funcionalidad y esfuerzo cuando se trata sistemas Cliente – Servidor y Sistemas Distribuidos (8).

Cosmic FFP, intenta de corregir el problema que se presenta en IFPUG Puntos de Función, pero al ser un método relativamente nuevo (9) y al no tener antecedentes de haber sido probado durante un tiempo extenso, es propenso a fallar en los resultados.

Dados estos antecedentes, uno de nuestros objetivos, es corroborar la investigación realizada por la PhD Lotte de Rore en la que define una nueva aproximación para medir y definir la productividad del Software, empleando un método de estimación que evalúa la aproximación tamaño/esfuerzo del desarrollo de software. COCOMO II es un ejemplo de un modelo de estimación (10). Este modelo será usado en nuestros casos de estudio.

Un modelo de estimación ayuda a establecer presupuestos y cronogramas o a realizar un análisis de desventajas en el desarrollo (10) que a simple vista un Jefe de Proyectos o Desarrollador no podría detectar. La estimación del volumen de trabajo requerido para un proyecto, se basa en las características más específicas del mismo, razón por la cual, el esfuerzo requerido para el proyecto puede servir como norma y/o punto de referencia para los proyectos futuros, lo que se convierte en la expectativa básica de un modelo de estimación.

Comparando la estimación del Esfuerzo necesario y el esfuerzo dedicado, se puede tener un indicador que muestre cuán productivo se es en el desarrollo de un proyecto.

1.3 Estructura del proyecto.

Podemos dividir el proyecto en 2 partes fundamentales:

En el capítulo 1 hablaremos del panorama general del proyecto y específicamente en el numeral 1.4 explicaremos el modelo de estimación que será usado para investigar el nuevo enfoque de productividad: COCOMO II. Este modelo además, será usado para validar el enfoque innovador que desemboca en un extenso caso de estudio, descrito en el capítulo 2.

El capítulo 2, describe la configuración del ambiente de medición con el modelo COCOMO II y se analiza los resultados obtenidos a lo largo del desarrollo del proyecto en las empresas participantes.

En el capítulo 3, se presentará el análisis del modelo utilizado, así como de los resultados del proyecto realizado en Ecuador. Este capítulo busca proporcionar a las empresas participantes, los puntos débiles y fuertes del Modelo utilizado y además, brindar sugerencias para mejorar la productividad.

1.4 COCOMO II (Constructive Cost Model)

En 1981, B. Boehm publicó COCOMO, su significado en inglés COnstructive COst MOdel; este modelo estima el número de hombres-mes que podría tomar desarrollar un producto de software. El primer modelo (11), fue nombrado COCOMO81 se desarrolló basado en el juicio experto y una base de datos de 63 proyectos de software terminados. No obstante, el desarrollo de software ha cambiado considerablemente desde que se introdujo este modelo: los proyectos siguen un modelo espiral evolutivo o un modelo de desarrollo en lugar de un modelo de cascada como asume COCOMO81, la complejidad de los proyectos de software ha aumentado cada vez más y hablamos de proyectos comerciales que usan componentes COTS. Como respuesta a estas evoluciones, el modelo constructivo de costo fue revisado y se convirtió en COCOMO II (10).

COCOMO II se compone de 4 modelos: el modelo de Puntos de Objetos o modelo de Composición de Aplicaciones, el modelo de Reuso o modelo basado en componentes, el modelo de Diseño Inicial o Temprano y el modelo Post-Arquitectura. El modelo de Puntos de Objeto o de Composición de Aplicaciones puede utilizarse como una métrica para composición de aplicaciones. Su estimación se basa en el número de pantallas, informes y componentes 3GL, que en su mayoría se ve en las fases o ciclos espirales que generalmente incluyen prototipado. Al final se

tendrá una mejor estimación para este tipo de esfuerzo que generalmente son desarrollados con lenguajes de tercera generación.

El modelo de Reuso o Basado en Componentes, en COCOMO II, usa un modelo no lineal para estimar el tamaño del software cuando éste incluye componentes reusables. El análisis de 3.000 casos de reuso de módulos realizado en el laboratorio 38 de Ingeniería de Software de la NASA indica que el costo asociado al reuso es una función no lineal debido a dos razones (12):

- Existe un costo base, de alrededor de un 5%, que contempla la evaluación, selección, y asimilación del componente reusable.
- Pequeñas modificaciones generan desproporcionadamente grandes costos. Esto se debe al esfuerzo por comprender el software a ser modificado, testear y chequear las interfaces.

El modelo de diseño inicial o temprano es un modelo de diseño de alto nivel; éste podría ser usado en la etapa de diseño arquitectónico para explorar alternativas de arquitectura o las estrategias para el desarrollo incremental. Este modelo es más cercano al modelo COCOMO original. El modelo Post-Arquitectura, por otro lado, es un modelo más detallado que se puede utilizar en las etapas de desarrollo y mantenimiento. Es la versión más detallada de COCOMO II.

Tanto el modelo de diseño inicial y el modelo de la Post-Arquitectura utilizan la misma fórmula para calcular la cantidad de esfuerzo necesario

para desarrollar un proyecto de software. En el resto de esta sección nos centraremos sólo en el modelo Post-Arquitectura.

1.4.1 El modelo

Fórmula

El modelo COCOMO II utiliza la medición de tamaño del software y un número de manejadores de costo (factores de escala y multiplicadores de esfuerzo) para estimar la cantidad de esfuerzo requerido para desarrollar un proyecto de software. La estimación del esfuerzo es expresada en personas-meses (personas por meses) y podría ser obtenido con la siguiente fórmula.

$$PM = A \cdot Size^{E} \prod_{i=1}^{n} EM_i$$

Ecuación 1: Esfuerzo

donde

$$E = B + 0.01 \cdot \sum_{j=1}^{5} SF_j$$

Ecuación 2: Exponente E

En esta fórmula A y B son factores constantes, los valores para éstos dos parámetros fueron obtenidos en la calibración de 161 proyectos en las base de datos de COCOMO II (10) y son inicialmente iguales a 2.94 y 0.91 respectivamente. En el exponente de la fórmula se encuentra 5 factores de escala (SF) que servirán

para ajustar las economías o deseconomías de escala que se presentan en proyectos de software de diferentes tamaños. Cuando el exponente E es inferior a 1, se tendrá una economía de escala. Esto significa que cuando el tamaño del proyecto se duplica, el efecto sobre el esfuerzo será menor que el doble. Sin embargo, cuando el exponente E es mayor que 1 el proyecto muestra una desaeconomía de escala lo que doblaría el tamaño y podría provocar un aumento de más del doble en el esfuerzo. El exponente E consiste en 5 factores de escala: Precedencia, Flexibilidad de Desarrollo, Resolución de Riesgos y Arquitectura, Cohesión del Equipo y Maduración del Proceso. Los factores de escala son definidos de acuerdo al nivel de cada proyecto, cada uno tiene un rango de niveles de calificación muy baja a muy alta. Cada nivel de calificación tiene un peso. Este peso se determina utilizando inicialmente la base de datos de COCOMO II de 161 proyectos. Inicialmente, estos proyectos se utilizaron para determinar los valores usando regresión múltiple (13), después un análisis bayesiano (14) fue usado para calibrar los pesos iníciales de los parámetros.

Los multiplicadores de esfuerzo, son características del proyecto que tienen un efecto lineal en el esfuerzo de un proyecto. El modelo

post-arquitectura define 17 multiplicadores de esfuerzo, similar a los factores de escala, los multiplicadores de esfuerzo tienen un rango de niveles definido, cada uno con un peso inicial definido con la base de datos de COCOMO II de 161 proyectos. Cada multiplicador de esfuerzo con excepción de "Planificación del Desarrollo", podría ser considerado como un módulo individual. Nosotros podríamos dividir los multiplicadores de esfuerzo en varias clases:

Factores del Producto (fiabilidad del software, tamaño de la bases de datos, complejidad del producto, reusabilidad en el desarrollo y documentación), Factores de la Plataforma (volatibilidad de la plataforma, limitación en el tiempo de ejecución, limitación de almacenamiento principal), Factores del personal (capacidad de análisis, capacidad del programador, continuidad del personal, experiencia en la aplicaciones, experiencia en la plataforma, experiencia en el lenguaje de programación y en las herramientas).

Medición del tamaño

COCOMO II es basado en el método de líneas de código (LOC) a pesar de que existen muchas paradojas referentes a este tema. Si se toman en cuenta algunos lineamientos para contar el tamaño se obtendrá una respuesta aceptable para el cálculo en el modelo de estimación. COCOMO II solo usa el dato del tamaño que influya

en el esfuerzo, esto incluye el código nuevo, copiado y modificado. El tamaño es expresado en "miles" de líneas de código (KSLOC). Con la finalidad de definir que es una línea de código, el modelo usa una lista de verificación definida por el Instituto de Ingeniería de Software (SEI) (15).

Como se había dicho antes, la medición del tamaño en la fórmula de COCOMO II, incluye todos los datos que influyen en el esfuerzo del proyecto. Sin embargo, la reutilización y el código modificado no pueden ser contados de la misma forma que un código nuevo, por lo tanto la fórmula puede ser usada para realizar las diferentes cuentas equivalentes con el fin de agregar todas las variantes en una sola medición del tamaño del proyecto.

El equivalente a KSLOC puede ser calculado con la siguiente fórmula (16):

$$Equivalent\ kSLOC = Adapted\ kSLOC\ \cdot \left(1 - \frac{AT}{100}\right) \cdot AAM$$

Ecuación 3: KSLOC Equivalente

donde:

$$AAF = (0.4\ \cdot DM\ ) + \ (0.3\ \cdot CM) + \ (0.3\ \cdot IM)$$

Ecuación 4: Esfuerzo para modificar código existente

y

$$AAM = \begin{cases} \dfrac{AA + FF \cdot (1 + (0.02 \cdot SU \cdot UNFM))}{100} & for\ AAF \leq 50 \\ \dfrac{AAF + AA + (SU \cdot UNFM)}{100} & for\ AAF > 50 \end{cases}$$

Ecuación 5: VAriable de fórmula KSLOC Equivalente

En estas fórmulas, AT es la cantidad de código generado automáticamente por cualquier tipo de herramienta que se esté utilizando para el desarrollo. DM es el porcentaje del modelo que ha sido modificado. CM es el porcentaje de código que ha sido modificado. IM representa el porcentaje de esfuerzo necesario para integrar el código adaptado o reutilizado. Se puede observar que la cantidad de esfuerzo para modificar el software existente no sólo es en función de la cantidad (AAF), pero también depende de la comprensibilidad del software existente (SU) y de la relativa falta de familiaridad de programador con dicho software (UNFM). AA representa el esfuerzo que se necesita para decidir si un módulo reutilizado es apropiado para ser utilizado en la aplicación.

De ésta fórmula, se puede ver que el software que es reutilizado sin ninguna modificación puede ser válido como un equivalente de SLOC, esto se debe al esfuerzo necesario para decidir si el módulo es adecuado para la reutilización (AA) y el esfuerzo necesario para integrar el software reutilizado en el producto en conjunto (IM).

Con estas directrices para medir el tamaño de un producto desarrollado, se podría estimar el esfuerzo requerido para desarrollar un proyecto de software.

1.4.2 COCOMO II como medida de la productividad.

COCOMO II estima la carga de trabajo en un proyecto, de ser necesario, tomando en cuenta características específicas.

El modelo COCOMO II podría ser usado como una medida de productividad cuando esta estimación es usada en el marco de la productividad del proyecto frente a la productividad calculada por COCOMO II, comparando el esfuerzo estimado con la carga de trabajo real del proyecto. Cuando un proyecto lleva más tiempo de lo prescrito por COCOMO II, el proyecto podría ser menos productivo que cuando el proyecto lleva menos tiempo que lo prescrito por COCOMO II; en otras palabras utilizamos la estimación de COCOMO II como norma para la medición de la productividad y comparar todos los proyectos con esta norma.

Sin embargo, COCOMO II está diseñado para realizar estimaciones. Esto significa que las directrices para la medición de líneas de código se definen con respecto al esfuerzo que más o menos se necesita. Por ejemplo, el recuento de las líneas reutilizadas: estas líneas de código se transforman en líneas

equivalentes de código (“líneas de código normales”) con el fin de cuantificar el esfuerzo que se necesita para crearlas.

Lo importante es el marco de referencia que se desea para el proyecto.

En consecuencia, para tener una buena comparación con la norma, la medición (es decir, la estimación del modelo) debe ser lo más próximo posible a lo que prescribe el modelo. Por lo tanto, todas las directrices que se indiquen en el modelo necesitan ser estrictamente seguidas.

El modelo COCOMO II es un modelo paramétrico. Una vez que la información sobre los propios proyectos se encuentre disponible, el modelo puede ser calibrado de acuerdo a la situación actual de la empresa.

1.4.3 COCOMO II como medida de la productividad.

Una de las principales fortalezas del modelo COCOMO II (10) es la extensa lista de controladores de costo (multiplicadores de esfuerzo y factores de escala).

Estos controladores de costo capturan características del desarrollo de software que afectan al esfuerzo que se necesita para completar el proyecto. Por lo tanto, estos factores ofrecen una información verdaderamente útil. Al observar las calificaciones de los

controladores de costo para cada proyecto se puede indicar que características del proyecto tuvieron o no una influencia en el esfuerzo estimado. La apreciación de ese análisis puede ser comunicada en los informes de gestión. Como tales, estos informes son un instrumento de gestión que da una indicación de los parámetros que necesitan atención dentro de la empresa con el fin de mejorar la productividad. En el resto, nos centramos únicamente en los multiplicadores del esfuerzo y no en los factores de escala.

Como se explicó anteriormente, el valor real del efecto de un controlador de costos se determinó mediante el uso de los 161 proyectos en la base de datos de COCOMO II. Como hay muy poca información disponible acerca de estos 161 proyectos, podemos asumir que los valores calculados son reales. Sin embargo, éstos podrán ser diferentes para los proyectos que nosotros analicemos. Por lo tanto, es aconsejable una calibración inicial del modelo para determinar si los valores son adecuados para el medio ambiente en el que se desarrollará el proyecto. En la siguiente sección describiremos los informes que podrían ser producidos para identificar el efecto real de un multiplicador de esfuerzo en los datos del proyecto.

1.4.4 Reportar multiplicadores de esfuerzo.

Idea inicial

Los multiplicadores de esfuerzo no solo constituyen una parte importante en el cálculo del esfuerzo, sino que además la puntuación que se le otorgue a cada multiplicador nos ayudará a analizar que áreas, dentro del proceso de desarrollo de software, necesitan mayor atención para que con acciones correctivas, la productividad mejore. Para lograr esto se plantea la construcción de un gráfico en el que la productividad relativa se ve reflejada frente a la calificación de cada multiplicador de esfuerzo, éste nos ayudará a identificar, tanto los proyectos productivos como los no productivos, las puntuaciones que se dan con mayor frecuencia para cada multiplicador. Cuando nos referimos a productividad relativa hablamos del resultado de la división del esfuerzo estimado dividido para el esfuerzo real. Un proyecto es calificado como productivo cuando su productividad relativa es mayor a 1, mientras que un proyecto es calificado como no productivo cuando su productividad relativa es menor a 1.

Una explicación práctica de la idea inicial, se presenta en el gráfico 1.1; en este gráfico podemos apreciar cómo se graficaron la productividad relativa frente a cada posible valor que puede tomar un multiplicador de esfuerzo, que para el ejemplo lo llamaremos

$EM_{i,}$ es decir, cuando EM_i toma el valor de N (normal) su productividad relativa es igual a 1, si descendemos al siguiente valor, por ejemplo a L (Low, bajo) podemos notar que la productividad relativa también reduce su valor y así sucesivamente, lo cual, para éste ejemplo, nos indica que mientras más baja es la calificación del multiplicador de esfuerzo la productividad relativa también lo será, y además notaremos que es conveniente centrar la atención y desarrollar acciones de mejora en las áreas en las que se vea involucrado el multiplicador de esfuerzo en análisis. En un caso ideal, productividad relativa es igual a 1, el gráfico debería presentarse con todos las líneas verticales alineadas en 1.

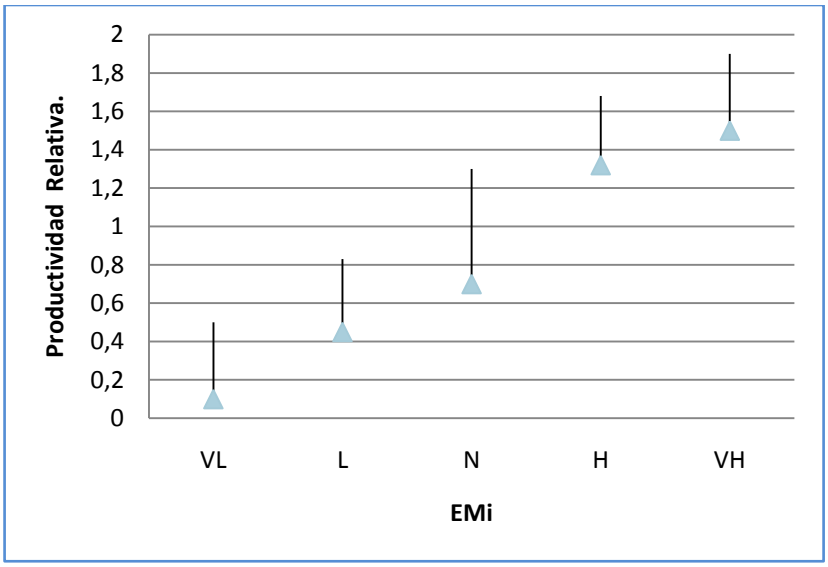


Ilustración 1: Caso Ideal de Productividad Relativa

Adicional a lo expuesto en el párrafo anterior, Lotte De Rore, autora del estudio realizado en Bélgica, hace referencia a un informe adicional denominado: empresas dependientes del valor de los multiplicadores de esfuerzo. Este reporte le permitió investigar cual es la influencia de un multiplicador de esfuerzo específico en el esfuerzo resultante de los proyectos de la empresa, esto mediante el cálculo del ESFUERZO NORMALIZADO, el cual permite obtener un valor del esfuerzo solo con la influencia del multiplicador de esfuerzo que se desee analizar, es decir, que los valores restantes quedan neutralizados en este cálculo; esto se reduce a la siguiente fórmula:

$$Esfuerzo\ normailzado_{EM_i} = R \cdot Size^E$$

Ecuación 6: Esfuerzo Normalizado

Donde el valor de R representa la pendiente del gráfico del esfuerzo normalizado, y que además es directamente proporcional al valor inicial del multiplicador de esfuerzo. De esta manera se determina el modo de cuantificar la influencia del multiplicador de esfuerzo en el o los proyectos incluidos en el análisis.

1.4.5 Evaluación del modelo COCOMO II

El modelo COCOMO II es algo más que un simple modelo. No es necesario un entrenamiento previo para realizar la medición. Una vez establecido un buen procedimiento, contar las líneas de código se vuelve algo trivial. Sin embargo, la dificultad está en determinar los valores correctos y exactos para los manejadores de costo, aunque el modelo da una explicación por cada manejador de costo, aún hay una cierta subjetividad a la hora de determinar los valores para cada uno de ellos, una apreciación errónea de alguno de los manejadores de costo, podría tener una influencia significativa en el cálculo de la estimación (17).

La desventaja principal del modelo COCOMO II, es el uso de líneas de código como medida del tamaño del software, pues existen un sinnúmero de paradojas en cuanto a este tema (18). Aunque el modelo solo incluye líneas de código fuente y da una descripción acerca de cómo incluirlas, se pueden presentar paradojas, como por ejemplo el uso de diferentes lenguajes y/o estilos de programación. Cuando se implementa éste modelo se debe ser consciente de esto y de ésta forma saber cómo interpretar los resultados.

Con respecto al uso de este modelo como medida de la productividad, se podría usar los valores de los manejadores de costo para compararlos con los de otras empresas, y en particular

con los de los 161 proyectos usados para calibrar éste modelo. Como tal, este modelo es visto como la norma con la que un proyecto podría ser productivo. Sin embargo, como la mayoría de los modelos paramétricos, podría ser interesante realizar una calibración con los proyectos de la propia empresa. En este caso, el modelo consiste, además de los parámetros A y B, de 17 multiplicadores de esfuerzo y 5 factores de escala. Lo significa que cuando se realiza la regresión múltiple sobre log (esfuerzo) (13), hay un conjunto de datos de al menos 120 puntos. Sin embargo la primera calibración de los factores constantes A y B en el modelo, podría ser realizada con menos datos. Aunque, el modelo podría dar un marco referencial que defina la productividad de un proyecto, la principal fuerza del modelo COCOMO II es su extensa lista de características del proyecto (multiplicadores de esfuerzo y factores de escala), las cuales tienen una influencia directa en el desarrollo de software, pero adicionalmente, ésta puede ser cuantificada. Es así como los manejadores de costo, indican los puntos de especial interés para una posible mejora de la productividad.

CAPÍTULO 2

2. MEDICIÓN DE PRODUCTIVIDAD: CASOS DE ESTUDIO

2.1 Escogiendo el método de medición.

Nuestro objetivo, con este proyecto, es la creación de un entorno para la detección y la medición del desempeño del departamento de Desarrollo de Software de las empresas participantes.

Si bien, un proyecto a gran escala tendría la intención de buscar las tendencias en las TICs de toda el área de desarrollo, para este proyecto solo se realizarán análisis individuales de proyectos de Software. Esto, sumado a las condiciones y solicitudes de las empresas participantes son los parámetros que nos dieron la pauta para elegir el método de medición adecuado. Además, el tiempo y el esfuerzo necesario para recoger los datos deben mantenerse al mínimo y no crear una sobrecarga de trabajo.

Como resultado, de la elección del método de medición, se prefieren aquellos que ofrezcan la oportunidad de automatizar dicho proceso, adicionalmente el método debe tener una técnica que permita la evaluación comparativa de los resultados con otras empresas.

Es importante aclarar, que las mediciones pueden ser realizadas en el momento de la terminación del proyecto, pues generalmente es en este punto donde hay más información disponible.

De los tres métodos en consideración: IFPUG Function Points, Cosmic Full Function Points, y COCOMO II (ver tabla 2.1), el escogido fue el último citado debido a que utiliza las “Líneas de Código” como la medida de tamaño

las cuales podrían ser contadas fácilmente de manera automática. Sin embargo existen muchas paradojas en cuanto al conteo de las líneas de código (19). Aún así, si el ambiente inicial es asignado correctamente y las reglas de cómo contar las líneas de código son indicadas correctamente a los desarrolladores, el modelo puede ser considerado teóricamente y matemáticamente más correcto que el método de Function Points.

Para Cocomo II, el tamaño del proyecto es tomado desde el punto de vista de la implementación. Esto contrasta con los otros métodos a consideración, donde el tamaño del proyecto es tomado desde el punto de vista del usuario. Por supuesto que para los fines de este proyecto es mucho más conveniente tomar las medidas de productividad desde la perspectiva del desarrollador ya que esto forma un punto de vista mucho más interesante y que además puede brindar mayor información que el punto de vista del usuario.

La facilidad de medición fue el factor decisivo que llevó a COCOMO a ser el método escogido, su principal punto negativo son las paradojas en las líneas de código, aunque pueden ser sobrellevados si la inducción inicial se realiza correctamente.

	IFPUG FP	Cosmic FFP	COCOMO II
Medición del tamaño	Tamaño funcional	Tamaño funcional	Basado en líneas de código
Datos Históricos/Análisis Comparativo	Base de datos ISBSG	Base de Datos ISBSG, poca cantidad de datos.	Multiplicadores de Esfuerzo y Factores de Escala
Puntos de Vista	Perspectiva del Usuario	Perspectiva del Usuario	Perspectiva del Programador
Facilidad de Medición	Conteo Manual, se necesita entrenamiento	Dificultad de automatización. Se necesita entrenamiento	Conteo de Líneas Automático

Tabla 1: Comparación de Modelos

2.2 Preparando el ambiente de medición.

Habiendo optado por COCOMO II, el siguiente paso es analizar los factores críticos, investigar los puntos determinantes para concluir si la medición con COCOMO II es viable en una empresa en particular. La sección 2.2.2 describe como pueden obtenerse las puntuaciones de los multiplicadores de esfuerzo y factores de escala. La sección 2.2.3 describe como el tamaño, llamado líneas de código, podría ser medido y como enfrentar las diferentes paradojas que implica este conteo.

2.2.1 Factores críticos

La fórmula de COCOMO II usa las líneas de código, los factores de escala y los multiplicadores de esfuerzo para estimar el esfuerzo requerido para desarrollar una parte de un software. Este esfuerzo es expresado en personas-mes y podría ser obtenido con las fórmulas 1 y 1.1, donde A y B son factores constantes, A=2.94 y B=0.91 para COCOMO II, mientras EM representa los multiplicadores de esfuerzo y SF los factores de escala. El tamaño del software es expresado en KSLOC.

Con el fin de hacer una evaluación correcta de la productividad, es decir, una comparación correcta entre la carga real del proyecto y la calculada por COCOMO II, es importante contar las cosas exactamente según lo prescrito por el modelo COCOMO II. Hay tres puntos importantes a considerar: un registro de tiempo correcto, un conteo correcto de las líneas de código y una relación correcta entre el registro del tiempo y las líneas de código.

Registro del tiempo

Para el registro de tiempo, es importante que se cuente sólo la carga de trabajo que se incluye en el modelo COCOMO II. Según COCOMO II, la fase de planificación y la fase de requerimientos no tienen que ser contadas como parte del esfuerzo de desarrollo. Para el caso de las empresas participantes, el tiempo será registrado en la fase de desarrollo de proyectos. Es importante tomar en cuenta que cada desarrollador, perteneciente a un equipo, tiene a cargo uno o dos proyectos.

Además de la identificación de las tareas, en donde se incluye el registro del tiempo, también necesitamos el número de horas trabajadas específicamente en cada tarea. En ciertos casos, las empresas pueden incluir en sus políticas el uso de herramientas de planificación en las que cada desarrollador pueda registrar las horas invertidas en el proyecto que se encuentre trabajando. Ninguna de las empresas participantes usa herramientas de este tipo, pero es importante mencionar que los datos de registro de tiempo podrían ser tomados mediante este método siempre y cuando sean obtenidos dentro de la empresa para garantizar la confiabilidad de los datos extraídos de la herramienta.

Contando las líneas de código

Las líneas de código son necesarias para la medición y además son el segundo punto crítico. Los detalles de la implementación de un conteo automático de líneas de código serán explicados en la sección 2.2.3. En este capítulo hablaremos sobre la manera correcta de contar las líneas de código a fin de coincidir con el registro de tiempo del proyecto, con esto nos referimos a que tanto las líneas de código como el registro del tiempo deben ser tomados en forma paralela, de esta forma evitamos que el conteo sea incorrecto.

Además de establecer el momento oportuno de contar, también necesitaremos un inventario de todos los módulos que fueron creados y modificados durante el proyecto. Finalmente, es importante tratar correctamente la parte de la reutilización de código pues se trata de una cuestión importante y además implica resolver algunas de las

paradojas conocidas de las mediciones de la productividad con las líneas de código. Los detalles de cómo hemos tratado con la reutilización de código se explican por separado en la sección 2.2.3.

Adaptación del esfuerzo y el tamaño

Un recuento correcto de las líneas de código y un registro correcto de la carga de trabajo no es suficiente: también es necesario una adaptación correcta entre estas líneas de código y el tiempo de registro.

Bajo ningún motivo deben existir líneas de código de los equipos no que no registren tiempo. Del mismo modo, ningún registro de tiempo debe incluir líneas de código no entregadas o ningún registro de tiempo de tareas puede ser contado si las líneas de código no fueron contadas.

Para evitar este problema, se ha diseñado una hoja de registro de datos. Este registro es el que usarán los desarrolladores y demás involucrados en esta fase.

2.2.2 Factores de Escala y Multiplicadores de Esfuerzo.

Para cada proyecto, los manejadores de costo (factores de escala y multiplicadores de esfuerzo) necesitan ser puntuados entre las categorías “muy bajo” y “extra alto”. Con la finalidad de establecer esta puntuación de la manera más objetiva posible, adjuntamos la tabla 2.2 en la que se muestra cada factor con una breve descripción (10). Por medio de una encuesta, la cual será realizada por los

participantes una vez concluido el proceso de pre-prueba, se comprobó la claridad de las preguntas. En la pre-prueba fueron evaluados 2 proyectos. Se solicitó al equipo de trabajo de ambas empresas involucrados en el proyecto que llenase las 3 tablas preparadas para la toma de datos, además se elaboró un Plan de Ejecución de pre-pruebas que consiste en una guía completa y explicativa del proceso.

Algunos factores de costo, se subdividen en múltiples criterios (10). Por ejemplo, la complejidad del producto se subdivide en las operaciones de control, operaciones de cálculo, dispositivo de las operaciones a su cargo, las operaciones de gestión de datos y la interfaz de usuario de las operaciones de gestión. Para cada criterio se incluye una pregunta.

2.2.3 Líneas de código

Con la finalidad de construir una buena base de referencia para tener un cálculo correcto de la productividad, todas las entradas en la Fórmula de COCOMO II deben ser las más correctas posibles.

Por lo tanto, en la medida de lo posible, las directrices para contar las líneas de código que se describen en el libro de COCOMO II (10), una de las principales preocupaciones son las paradojas inherentes al uso de LOC como la medida de tamaño, según lo indicado por Jones (19). Debe tenerse en cuenta la creación de un ambiente previo a la toma de datos, para descartar estas paradojas y a fin de evitar conclusiones erróneas extraídas de los resultados de la medición.

En las secciones siguientes, se explica las medidas que se tomaron para descartar estas paradojas. Como observación general, se debe notar que una solución imperfecta puede ser suficiente, siempre que las paradojas se resuelven en gran medida.

2.2.3.1 Resolución de paradojas resultantes de la utilización de diferentes lenguajes de programación.

Una línea de código escrito en un lenguaje de programación no tiene el mismo valor que una línea escrito en otro lenguaje de programación. Para nuestro caso, no tendremos ningún tipo de problema, los proyectos escogidos para la medición están escritos en un mismo lenguaje tanto del lado del cliente y del servidor. Es una total coincidencia que ambas empresas usen lenguajes similares.

2.2.3.2 Resolución de paradojas resultantes de las líneas de código nuevas frente a las líneas de código modificadas

De acuerdo a COCOMO II, los módulos modificados tienen que ser contados de una manera diferente que los módulos nuevos. No todas las líneas de código de los módulos modificados tienen que ser contadas, solo deben serlos las líneas modificadas. Como se había mencionado antes, en la fase de pre-prueba se creó la "*tabla de datos*

para conteo de líneas de código, la tabla está constituida en dos partes: la cabecera y el cuerpo. En la Cabecera se especifica el nombre del proyecto, un código único, entre otra información. Este código servirá para identificar las líneas modificadas en el código, es decir si una línea es modificada se deberá colocar al final de ésta, el código del proyecto, esto nos ayudará a verificar la atribución de la línea al proyecto correcto.

Adicional a esto, en una de las empresas existe la ventaja del uso de SVN y el hábito de documentar las líneas que se cambian cada vez que se realiza una modificación. En el caso de la otra empresa, se empezó a introducir el hábito de la documentación aunque de todas formas, ambas empresas usaron las tablas de datos introducidas en la pre-prueba.

2.2.3.3 Resolución de paradojas resultantes de entregas múltiples de la misma pieza de código

Muchos de los proyectos en las empresas participantes no se entregan todos a la vez, las entregas se dan en diferentes etapas. Si un módulo es entregado en etapas múltiples, y cada vez con algunas modificaciones, más líneas de código modificadas deben ser contadas, y así con cada entrega, por lo que resultaría mejor contar las líneas de

código solo en la entrega final y no cada vez que se haga una entrega previa.

Se llegó a un acuerdo con las empresas participantes, éste consiste en que solo las entregas finales serán contabilizadas más no las parciales. Hay que tener en cuenta que la fecha marcada como final no siempre corresponde a la fecha en que se realizó la entrega, puesto que después puede haber modificaciones que también deben ser contabilizadas. De esta forma ahorraremos tiempo en la contabilización y nos aseguramos que las líneas serán contadas cuando el módulo esté terminado.

2.2.3.4 Resolución de paradojas resultantes de diferentes estilos de programación

Aunque algunas de las paradojas con respecto a las líneas de código ya han sido contrarrestadas mediante el uso de las tablas introducidas en la pre-prueba, resta tratar otras paradojas importantes. Un desarrollador que escribe mucho código de manera no estructurada parece ser más productivo que un programador que escribe su código de forma estructurada, a pesar de que éste último tendrá cualitativamente un mejor código. Si bien los desarrolladores de las empresas participantes han trabajado juntos durante algunos meses, existe diferencia en su estilo de programación, lo cual puede influir

considerablemente al momento del conteo. De esta forma llegamos al punto en que las empresas de alguna manera deben estimular a los desarrolladores a mantener una uniformidad en el estilo de programación. Para resolver este problema acordamos que desde la aceptación del proyecto hasta la finalización de la pre-prueba, los desarrolladores de alto nivel (Jefes de Desarrollo) revisarían el código de los Desarrolladores Junior a su cargo. Con la finalidad de que estos últimos adopten un estilo más uniforme de programación y que el conteo de LOCs tenga un margen de error inferior. Una regla impuesta por los Jefes de Desarrollo es reducir al mínimo el código muerto (funciones, propiedades, parámetros, variables que nunca se utilizan).

Por último, la reutilización de código también puede dar lugar a una paradoja, que es la más difícil de tratar, por lo que la trataremos por separado en el siguiente punto.

2.2.3.5 Resolución de paradojas resultantes de la reutilización de código frente a nuevo código

La reutilización de código es un problema importante y difícil de aplicar. El modelo COCOMO II está claramente diseñado para las estimaciones, lo que significa que las directrices para la medición de líneas de código se definen con respecto a los esfuerzos que más o menos se

necesitan. Los Módulos reutilizados no se pueden contar como código normal (recién escrito), pero sería un error no contarlos. Después de todo, el tiempo que es dedicado para decidir si un módulo será utilizado también forma parte del tiempo que se está invirtiendo en desarrollar un nuevo módulo. En el modelo COCOMO II, las líneas de código reutilizadas se transforman en "equivalente de nuevas líneas de código" con el fin de cuantificar el esfuerzo que sería necesario para crear estas líneas de código.

Es importante indicar que un proyecto que reutilizó algún código, tiende a ser más productivo que un proyecto que no lo utilizó (a pesar de que el primero tenga más líneas físicas de código), esto debido a que se tiene la misma salida (líneas de código) pero la funcionalidad es entregado con menos entrada (esfuerzo).

Sin embargo, utilizando el modelo COCOMO II como una medida de productividad, se hace una comparación entre el volumen de trabajo estimado y el esfuerzo real, en lugar de comparar la salida con la entrada. Por lo tanto, lo que es importante es el marco de referencia que deseemos en nuestro proyecto.

Con el fin de tener una buena comparación con la norma, nosotros estimaremos el esfuerzo normativo tal como lo indica COCOMO II esto es: calculando el esfuerzo tan

estrechamente cercano como sea posible. Hay que tomar en cuenta que en las empresas participantes existe una política de código reusable, está prohibido reescribir código. Si un módulo existe, éste deberá ser reusado; no existe objeción, esta política debe ser controlada por el Jefe de Desarrollo.

Siguiendo las directrices de COCOMO II, se debe emplear la siguiente fórmula (10):

$$LOC\ Equivalentes = LOC\ Reusadas \cdot \frac{(AA + 0{,}3 \cdot IM)}{100}$$

Ecuación 7: Líneas de código equivalentes.

Donde el AA representa el esfuerzo necesario para decidir que módulo es el apropiado para la reutilización. IM representa el esfuerzo necesario para integrar el software reutilizado en el programa final.

En una de las empresas participantes, en su mayoría, las librerías están siendo reutilizadas. Cada vez que se reutilizan, se realiza el mismo tipo de esfuerzo para integrar los módulos. Por lo tanto, se decidió utilizar los valores estándar para cada uno de los criterios de la fórmula. Para realizar un ejemplo sencillo asumiremos que el valor de AA será de 2, el valor para IM será de 10, es decir, sólo se necesita invertir el 10% de esfuerzo para integrar el software reutilizado en el programa y poner a

prueba el producto resultante, en comparación con la cantidad normal de integración y esfuerzo de prueba que se requeriría para construir un nuevo software de tamaño comparable (10). De esta forma, el código contará con un 5% de líneas reutilizadas.

En caso de que los componentes que puedan ser reutilizados sean muy grandes o se usen frecuentemente, se recomienda mantener una base de datos de los mismos en la que conste el número de líneas de código para evitar un re cálculo innecesario.

Por último, tenemos que considerar que la reutilización de los módulos puede ser recursiva: Un módulo puede usar a otro módulo y éste a su vez otros módulos más. En nuestras mediciones, hemos considerado la recursividad en un solo nivel de profundidad. Si el módulo A reusa el módulo B y el módulo B reusa el módulo C, el único módulo que es contado como "código normal" es el módulo A, el módulo B será contado como "código reusado" y el módulo C NO SERA CONTABILIZADO, esto debido a que a partir del módulo A no existe ninguna decisión o algún esfuerzo de integración para la reutilización del módulo C.

En plataformas cliente–servidor es posible detectar la reutilización mediante el uso de las sentencias

import/include/include_once, sin embargo una declaración de ésta clase no implica que todo el archivo será reutilizado, es probable que solo una parte se utilice. Esto significaría que se están contando demasiadas líneas de código que no deben ser contadas. Para cuidar esta parte, es necesario que se adopten medidas en las que solo un porcentaje de las líneas de código de los archivos importados, sean contadas. Vale la pena recalcar, que se instruyó a los desarrolladores para que las sentencias como: *import* * no sean incluidas en ninguna parte del código para no distorsionar los resultados.

2.3 Resultados de la Medición.

2.3.1 Calibración inicial del método.

El modelo COCOMO II ha sido creado con la ayuda de la opinión de expertos y de modelos estadísticos de datos de un conjunto de 161 proyectos para determinar los valores iniciales del modelo (14).

Al igual que la mayoría de métodos paramétricos, para mejorar la precisión, el modelo debe ser calibrado en su propio entorno (19) (20).

La empresa del caso de estudio 1 no cuenta con datos suficientes de proyectos anteriores que ayuden a iniciar una calibración efectiva del modelo COCOMO II, con esto nos referimos a que no existía un método establecido para el conteo de líneas de código, tampoco para selección de módulos para reutilización ni para la otorgar una

calificación a los diferentes manejadores de costo. Sin embargo, se nos otorgó la facilidad para realizar una pre-prueba en un proyecto que estaba en marcha, esto nos ayudó a inducir al personal y a familiarizarlo con el uso del modelo COCOMO II y de esta forma lograr que los datos finales sean tomados con mayor precisión.

2.3.1.1 Pre-prueba.

La pre-prueba es un proceso que duró 2 semanas, las cuales fueron contadas después de la sesión de 30 minutos de inducción donde se dio a conocer los detalles del proceso de medición de la productividad al personal involucrado. La pre-prueba tiene como objetivo validar las tablas de factores de escala, multiplicadores de esfuerzo y la tabla básica para la productividad con la finalidad de que se constituyan en herramientas para el cálculo de productividad usando el modelo de estimación COCOMO, independientemente del tipo de proyecto de software y de la empresa o grupo que lo desarrolle. De acuerdo al tipo de tabla, se necesitará la colaboración de los diferentes miembros del departamento de desarrollo de la empresa, entre estos: el Jefe del Grupo de Desarrollo, los Desarrolladores, el Jefe de Departamento de Desarrollo, entre otros. En el documento "Plan de Desarrollo de Pre-Prueba para la estimación de la productividad con COCOMO II." se puede obtener más detalles acerca del proceso.

2.3.1.2 Lecciones aprendidas de la pre-prueba y resultados del estudio.

El proceso de pre-prueba duró 2 semanas, en las cuales calculamos el esfuerzo requerido y validamos los datos obtenidos con el apoyo del Jefe de Desarrollo del proyecto quien emitió su opinión como experto.

A continuación presentaremos los problemas a los cuales nos enfrentamos durante la pre-prueba, los resultados obtenidos y las lecciones aprendidas.

Problemas encontrados:

1) La calificación otorgada por los desarrolladores en ciertos ítems de las tablas de multiplicadores de esfuerzo y factores de escala no estaba dentro del rango de valores permitidos que COCOMO II otorga.
2) Al igual que en el caso anterior, algunos ítems contaban con una calificación que no era la correcta según el criterio experto del Jefe de Proyectos.
3) La columna denominada "porcentaje de diseño modificado" de la tabla básica de productividad no estaba planteada correctamente, lo que complicaba el cálculo de las variables DM (design modified) y CM (code modified)
4) Las variables IM (integration effort) y AA (porcentaje de esfuerzo para decidir que módulo será el apropiado para reusar en la aplicación) no estaban planteadas apropiadamente, esto debido a que ningún documento que

les proporcionamos solicitaba esta información a los participantes.

Resultados:

Una vez revisados todos los inconvenientes relacionados a la toma de datos y a las tablas, procedimos a realizar una encuesta a los desarrolladores que utilizaron las tablas durante el proceso de pre-prueba. Esta encuesta tenía como objetivo medir que tan preparados estaban los participantes para la pre-prueba y, recibir retroalimentación del documento que se les fue otorgado como guía.

Como resultado de la encuesta obtuvimos la sugerencia de realizar una re-inducción a los participantes para afianzar lo aplicado durante la pre-prueba.

Otras observaciones que recibimos, nos permitieron mejorar las secciones de “Breve explicación” y “Explicación de los campos” de cada tabla.

Adicionalmente, para corregir las variables IM y AA descritas anteriormente diseñamos la tabla “Control de Módulos Reusados”.

Luego de aplicar todas las medidas correctivas para cada problema detectado en la pre-prueba, planificamos una nueva inducción al inicio de la etapa de toma de datos oficial, la cual arrancó una semana después de la realización de la pre-prueba.

Los resultados de KSLOC, esfuerzo, multiplicadores de esfuerzo y factores de escala se muestran en las siguientes tablas:

Proyecto	Esfuerzo	Ksloc	PLEC	FLEX	RESL	TEAM	PMAT
PS001	17,5	4,32	VH	H	L	H	VL
CF001	2,68	1,79	L	N	VL	N	VL

Tabla 2: Resultado de Pre-prueba y Factores de Escala

Proy	RELY	DATA	CPLX	RUSE	DOCU	TIME	STOR	PVOL	ACAP
PS001	VH	N	H	VH	VH	N	N	L	H
CF001	N	N	H	VH	VL	N	N	N	H

Tabla 3: Resultado de Pre-prueba - Multiplicadores de Esfuerzo

Proy	APEX	PLEX	LTEX	TOOL	SITE	SCED	PCAP	PCON
PS001	N	N	N	N	VH	N	H	L
CF001	H	VH	VH	VH	N	N	VH	H

Tabla 4: Resultado de Pre-prueba - Multiplicadores de Esfuerzo

2.3.2 Caso de estudio 1:

Una vez de haber tomado las medidas correctivas a los inconvenientes detectados en la etapa de pre-pruebas y finalizado el proceso de toma de datos oficial procedimos a realizar el cálculo y análisis para el caso de estudio denominado CF002.

2.3.2.1 Primeros resultados de la medición.

Durante el proceso de toma de datos para CF002, el entorno de desarrollo fue de relativa estabilidad, característica que años atrás no se había dado. Sin embargo, es válido acotar que este

proyecto tenía una cuota considerable de participación administrativa, lo que significa, seguramente que el esfuerzo real registrado difiera ligeramente de la realidad. Esto introduce una cierta incertidumbre a los resultados de la medición. A pesar de ello, debemos saber que los números que se obtuvieron de éste ejercicio, no son absolutos, si no que representan una primera medición que indicará las áreas que necesitan mayor investigación.

El tamaño de este proyecto fue de 3,05 KSLOC con un esfuerzo estimado por el método COCOMO II de 7,1 PM, considerando un valor de calibración de A=2,94 y B=0,91. Adicionalmente, solicitamos a la Katholieke Universiteie Leuven de Bélgica, se realice un cálculo de estimación de éste proyecto utilizando sus parámetros de calibración, el resultado que se obtuvo fue de 19,32 PM. Dados estos resultados y teniendo un valor de esfuerzo actual de 3,2PM podemos deducir que el proyecto CF002 es PRODUCTIVO comparado con las estimaciones de COCOMO II y KBC Bank.

Las razones por las que este fenómeno se dio durante este proyecto pueden ser explicadas en los siguientes puntos:

La cantidad de proyectos (2 en total, 1 por empresa) no es suficiente para garantizar un resultado confiable puesto que cada proyecto tiene una influencia considerable en el cálculo de los parámetros. La segunda posible causa podría enfocarse a la forma errada de calcular y/o registrar el tiempo y contar las

líneas de código, esto obviamente influye en el cálculo del esfuerzo actual. Esto generalmente ocurre cuando existe una sobrecarga de horas laborables (tiempo) y salto u omisión en el conteo de líneas de código que estuvieran desarrollándose o que ya habían sido desarrolladas y/o entregadas.

En tercer lugar tenemos que debido a que los proyectos a medirse tenían una arquitectura cliente–servidor la medición de estos variaba entre cada uno de los lados, es decir se consideran 2 proyectos diferentes, lo que implica que se deben determinar factores de conversión para de esta forma poder tener las líneas de código en una sola unidad medible, la determinación de este factor de conversión fue realizada en base a criterios de expertos que trabajan con los dos lenguajes de programación involucrados en los proyectos (JavaScript y Php), al ser esto una consideración absoluta se convierte en una posible causa de distorsión en los resultados.

Existen factores de entorno que también podrían influir en el resultado de las mediciones, por ejemplo: un desarrollador puede ser asignado a múltiples proyectos pequeños aumentando la probabilidad de cometer errores en los registros de los datos solicitados.

Proyecto	**Esfuerzo Actual (PM)**	**Ksloc**	**PLEC**	**FLEX**	**RESL**	**TEAM**	**PMAT**
CF002	3,2	3,05	L	N	N	H	VL

Tabla 5: Resultado Caso de Estudio CF002

Proy	RELY	DATA	CPLX	RUSE	DOCU	TIME	STOR	PVOL	ACAP
CF002	VL	N	VH	H	L	N	N	N	H

Tabla 6: Caso de estudio CF002 - Multiplicadores de Esfuerzo.

Proy	PCAP	PCON	APEX	PLEX	LTEX	TOOL	SITE	SCED
CF002	VH	H	H	N	L	H	L	H

Tabla 7: Caso de Estudio CF002 - Multiplicadores de Esfuerzo

2.3.2.2 Deduciendo áreas de mejora: resultados de la medición de multiplicadores.

Al tener solamente 2 proyectos, 1 por empresa, lo cual indica que no se tiene una cantidad suficiente para proporcionar la confianza suficiente para capturar el efecto correcto de un multiplicador de esfuerzo.

Debido a esta premisa, deduciremos las aéreas de mejora tomando en cuenta los resultados de los multiplicadores de esfuerzo del proyecto CF002 y los proyectos KBC.

2.3.2.2.1 Experiencia en el lenguaje y herramientas, LTEX

Este multiplicador de Esfuerzo representa, en una escala de “Muy bajo” a “Muy alto”, la experiencia que el grupo de desarrollo posee en el lenguaje de programación y herramientas utilizadas. Para el caso de

CF002 el valor asignado fue L (Low o Bajo). Esto en términos generales significa que la experiencia del grupo de desarrollo es menor o igual a 6 meses. Teniendo en cuenta, los resultados obtenidos en el análisis de éste multiplicador de esfuerzo para los proyectos KBC, en estos se indica que teniendo un valor de H (Higth o alto) se necesita solo un 86% del esfuerzo comparado con un valor de N (Nominal o normal) el cual necesita un 91%. Por lo tanto, es admisible considerar que para un valor de L, como en el caso de CF002, el esfuerzo necesario se elevaría a un 95%. Este valor nos indica que la empresa participante debe esforzarse por subir el valor de éste multiplicador de esfuerzo. Se recomienda analizar la forma en la que se componen los grupos de desarrollo, puesto que algunos pueden ser considerados débiles y otros fuertes por la experiencia que poseen.

2.3.2.2.2 Experiencia en la Plataforma, PLEX

Este multiplicador de esfuerzo representa, en una escala de “muy bajo” a “muy alto”, la experiencia previa que el grupo de desarrollo del proyecto tiene en la plataforma que utilizan. El valor asignado en este caso fue “normal o nominal”, lo que en términos generales significa que el grupo de desarrollo tiene 1 año de

experiencia de. En los proyectos KBC se los resultados fueron "alto" y "muy alto", puesto que la experiencia era entre 3 y 6 años. Para el caso de CF002, 1 año no es considerado tiempo suficiente para ser experto en un tema específico, por lo tanto el multiplicador de esfuerzo PLEX debe mejorar ya su valor inicial influye en el esfuerzo total en un 12%.

2.3.2.2.3 Tamaño de la Base de Datos, DATA

Este Multiplicador de esfuerzo captura el efecto de los requerimientos de una gran cantidad de datos de prueba sobre el producto en desarrollo. Dentro de los resultados de KBC se dan una variedad de resultados: bajo, normal y alto, lo que implica una serie de análisis y de opciones adicionales para reducir el esfuerzo en los proyectos que se ameriten. Para el caso de CF002 se obtiene un valor normal lo que significa se crea una cantidad mínima de datos de prueba y que además son creados por el propio desarrollador. Aunque para este caso DATA tiene una influencia mínima en el esfuerzo total, esta es relativa al tipo de proyecto, razón por la cual la empresa involucrada debe tener planificadas políticas que ayuden a reducir la influencia del esfuerzo si este multiplicador llegara a tomar un valor muy alto.

El incremento de la influencia puede ser de hasta un 37% según lo indicado en el análisis de KBC.

2.3.2.2.4 Restricción en el almacenamiento principal, STOR, y restricción en el tiempo de ejecución, TIME

Estos multiplicadores de esfuerzo están relacionados entre sí, pero además tienen algo en común con el multiplicador DATA y es que son muy dependientes del tipo de proyecto y de las exigencias del cliente. Para el caso de CF002 no existe ninguna restricción en particular, es por eso que el valor otorgado para ambos casos es "normal", pues no se necesita de ninguna atención extra en el tema de restricciones en tiempo de ejecución y almacenamiento principal. Sin embargo, al igual que para el multiplicador DATA se debe planificar qué medidas tomar en caso que los valores, para cualquiera de estos multiplicadores, se elevaran considerando que esto implicaría mayor esfuerzo en etapas ajenas al desarrollo, particularmente en el momento en que la arquitectura está siendo definida; de esta forma disminuye la probabilidad de que se presenten problemas en la etapa de desarrollo que luego impliquen un retroceso a etapas iniciales y por ende el esfuerzo se vea afectado directamente en un incremento importante.

2.3.2.2.5 Requerimiento de fiabilidad de software, RELY

Este multiplicador de esfuerzo es la medida de hasta qué punto el software debe realizar su función esperada durante un periodo de tiempo. Si el efecto de un fracaso es solo una molestia ligera el valor de RELY es de "muy bajo", pero si un fallo del sistema llegase a arriesgar vidas humanas, su valoración debe ser "Muy Alta".

Para el caso de CF002 se tiene un valor de "muy bajo", lo que significa que si el software llegara a fallar solo causaría un "pequeño inconveniente". Para KBC los valores fluctúan desde "muy bajo" a "normal". Para ambos casos se debe tomar en cuenta que, para llegar a obtener un valor "muy bajo" para RELY requiere más esfuerzo que un valor "normal", este fenómeno se presenta debido a que, durante la fase de desarrollo de la aplicación, se aplicaron medidas como, el uso de estándares de calidad altos, pruebas exhaustivas acorde al tipo de software entre otras; que requirieron un esfuerzo mayor comparado con un proyecto que no tomó estas medidas. Suponiendo que esta premisa en válida, dentro de una empresa el valor de RELY no debería variar significativamente entre proyectos ya que esto significaría que las medidas preventivas no fueron aplicadas correctamente.

2.3.2.2.6 Complejidad del producto, CPLX

Como indica su nombre, este multiplicador mide la complejidad del producto en desarrollo. CF002 tiene un valor de “extra alto” para este multiplicador, mientras que para KBC se tiene valores “bajos” y “normales”, aunque parezca contradictorio, si CPLX fue calificado como bajo implica una mayor cantidad de esfuerzo del que se hubiera tenido que realizar si se hubiera calificado como normal. Es importante recalcar que este multiplicador de esfuerzo se divide en términos de: operaciones de control, operaciones computacionales, operaciones de dependencia de dispositivos, operaciones de administración de datos y operaciones de administración de la interfaz del usuario.

Para el caso de los proyectos locales no se tiene una mayor apreciación de las tendencias debido a la falta de datos, pero KBC indica que es posible que esta contradicción sea la causa de la variedad de criterios sobre la que tiene que ser calificada.

2.3.2.2.7 Desarrollo para reusabilidad, RUSE

Este multiplicador de esfuerzo debe tener en cuenta el esfuerzo adicional necesario para desarrollar un módulo reutilizable. En CF002 este multiplicador obtuvo un

valor de “alto”. Si los valores fueran bajos se necesitaría el 24% menos de esfuerzo que si el valor fuera “muy alto” pero éste utilizaría un 37% adicional de un valor “normal”. La clave de la reusabilidad dentro de una empresa se centra principalmente en el equipo que realice el análisis técnico, pues éste determinará que módulo hará que la empresa sea una “multi-empresa”. Existen medidas que pueden mejorarse dentro de la empresa para pulir el proceso de reutilización y ésta disminuya el esfuerzo cuando un equipo de desarrollo decida utilizar un módulo antes desarrollado.

2.3.2.2.8 Documentación en el ciclo de vida, DOCU

Dada la naturaleza de las empresas locales, sería muy raro que la calificación para este multiplicador de esfuerzo esté más arriba de la calificación “normal”. Para el caso de CF002 el valor es “bajo”, lo cual quiere decir que tan solo algunas etapas están documentadas, no se especifica si las etapas documentadas ayudarán o no a disminuir/aumentar su influencia en el esfuerzo. Lo que se puede especificar claramente, y la empresa deberá tomar en cuenta, es que el LIDER de cada equipo de desarrollo es responsable de inspeccionar la documentación para garantizar que esté correcta; si esto

no se dá no es de extrañarse que los valores para este multiplicador lleguen a ser “muy bajos”.

2.3.2.2.9 Experiencia en la aplicación, APEX

Este multiplicador mide la experiencia del equipo de desarrollo en la aplicación. CF002 lo ha calificado como “alta” (3 años de experiencia), similar a lo recogido en KBC donde los valores están entre “normales” y “altos”, esto significa que un proyecto que tenga un valor de “alto” tendrá q utilizar un 11% menos de esfuerzo que un proyecto donde el grupo tenga una experiencia promedio de un año.

Es recomendable que la empresa se preocupe por planear qué medidas tomar cuando se integran nuevos miembros al equipo de desarrollo.

2.3.2.2.10 Uso de las herramientas de software, TOOL

Finalmente tenemos el multiplicador que mide el uso de herramientas de software, el cual tiene una calificación “alta”.

El porcentaje de esfuerzo que podría elevarse en caso de que la calificación disminuyera sería de un 6% según lo especificado por KBC, aunque para esta empresa este multiplicador no tenga una influencia muy significativa.

Para las empresas ecuatorianas analizadas se prevée un comportamiento similar, puesto que éstas invierten tiempo en la inducción del nuevo personal.

2.3.3 Caso de estudio 2:

De manera similar se realizaron los cálculos y análisis para el segundo caso de estudio denominado PS002.

2.3.3.1 Primeros resultados de la medición.

En la toma de datos para el caso PS002, a diferencia del caso CF002, se apreció el cambio en el ambiente de desarrollo, específicamente en el número de equipos que intervinieron en cada proyecto. Los dos equipos que participaron en PS002 mantienen métodos y formas diferentes de administrar las tareas. Adicionalmente, uno de los equipos se vio afectado por el cambio de sus integrantes durante tiempo de desarrollo. Debido a estos motivos los resultados de las mediciones pudieron presentar un margen de error.

El tamaño del proyecto de este caso de estudio fue de 2,03 KSLOC y la estimación por medio del modelo COCOMO II fue de 5,97 PM. Adicionalmente, solicitamos a la Katholieke Universiteie Leuven que realice el cálculo del esfuerzo con sus valores de calibración, obteniendo un esfuerzo de 20,35 PM. Con los resultados estimados contra el esfuerzo actual de 4,5

PM, podemos deducir que el proyecto PS002 es PRODUCTIVO bajo los cálculos estimados de COCOMO II y KBC Bank.

A continuación listamos los posibles motivos que generaron dichos resultados:

1) Poca cantidad de proyectos y por tanto su influencia fue alta en el cálculo de los parámetros de medición.
2) El nivel de organización del Jefe de Proyecto que debía atender solicitudes de mantenimiento de sistemas, por lo cual se veía obligado a cambiar las prioridades (baja, urgente, inmediato) de las tareas de los desarrolladores. Esto ocasionaba trabajos pendientes y pérdida de consistencia de la información recabada.
3) Falta de coordinación al momento de tomar los datos; es decir, que el desarrollador registra la información después de un tiempo prolongado de haber terminado una o más tareas, por lo que presumimos que los tiempos son inexactos.

2.3.3.2 Deduciendo áreas de mejora: resultados de la medición de multiplicadores.

Debido a que no es suficiente con un proyecto dentro de una misma empresa para el estudio del impacto del multiplicador en el esfuerzo, la deducción de cuáles deben ser las áreas que necesitan ser mejoradas se la realizará usando la información de los resultados del caso de estudio y la del KBC.

Proyecto	Esfuerzo Actual (PM)	Ksloc	PL EC	FL EX	RE SL	TEA M	PM AT
PS002	~4,5	2,03	N	L	L	N	VL

Tabla 8: Resultados Caso de Estudio PS002

Proyecto	RE LY	DA TA	CP LX	RU SE	DO CU	TI ME	ST OR	PV OL	AC AP
PS002	VL	H	N	L	N	N	H	L	H

Tabla 9: Caso de Estudio PS002 - Multiplicadores de Esfuerzo

Proyecto	PC AP	PC ON	AP EX	PL EX	LT EX	TO OL	SI TE	SC ED
PS002	L	L	N	H	H	N	VL	N

Tabla 10: Caso de Estudio PS002 - Multiplicadores de Esfuerzo

2.3.3.2.1 Experiencia en el lenguaje y herramientas, LTEX

El caso PS002 fue calificado con un valor de “alto”, lo que significa que el grupo de desarrollo cuenta con una experiencia aproximada de 3 años. Considerando los resultados de la investigación de este multiplicador en KBC, el esfuerzo para el caso PS002 es de 86%.

2.3.3.2.2 Experiencia en la plataforma, PLEX

Este multiplicador fue calificado con un valor de “alto”, lo que significa que el grupo de desarrollo cuenta con una experiencia aproximada de 6 años. En los proyectos del KBC se determinó que no había influencia de este multiplicador en la reducción del esfuerzo. El valor de

este multiplicador debido a su influencia es del 12% del esfuerzo.

2.3.3.2.3 Tamaño de la base de datos, DATA

El caso PS002 fue calificado con un valor de "Alto", lo que significa que hay gran cantidad de datos para realizar las pruebas representando un incremento del esfuerzo en un 23%, según los proyectos del KBC. Por tanto se recomienda que la empresa planifique políticas para reducir el esfuerzo con los datos de pruebas.

2.3.3.2.4 Restricción en el almacenamiento principal, STOR, y restricción en el tiempo de ejecución, TIME

Estos multiplicadores de esfuerzo están relacionados entre sí, pero además tienen algo en común con el multiplicador DATA y es que son muy dependientes del tipo de proyecto y de las exigencias del cliente. Para el caso PS002 la calificación de TIME fue "normal", mientras STOR fue "alto". Este último indica, según los resultados de KBC, un incremento del 37% de esfuerzo sobre lo "normal". Por lo tanto recomendamos reducir la influencia del multiplicador STOR.

2.3.3.2.5 Requerimiento de fiabilidad de software, RELY

Si el efecto de un fracaso es solo una molestia ligera, el valor de RELY es de “muy bajo”, pero si un fallo del sistema llegase a arriesgar vidas humanas, su valoración debe ser “muy alta”.

Para el caso PS002 este multiplicador se lo calificó con un valor de “muy bajo” lo que significa que de ocurrir un fallo el impacto de éste sería leve.

2.3.3.2.6 Complejidad del Producto, CPLX

Este multiplicador representa la complejidad del producto. Para el caso PS002 obtuvo una calificación de de “normal”. Mientras que un proyecto con la calificación de “bajo” necesita un 67% de esfuerzo adicional comparado con otro de valor “normal” (según el análisis de los proyectos de KBC). Es importante acotar, que no se proporcionó información adicional que pudiera determinar si los criterios usados fueron los más adecuados para calificar este multiplicador.

2.3.3.2.7 Desarrollo para reusabilidad, RUSE

Este multiplicador de esfuerzo debe tener en cuenta el esfuerzo adicional necesario para desarrollar un módulo

reutilizable. Para el caso PS002 se lo calificó con un valor de “bajo”, Lo que según los proyecto del KBC representa un esfuerzo de 24% menos en relación a lo “normal”, según las evaluaciones para este multiplicador en el KBC. Esta empresa participante tiene una política de re-utilizar todo lo generado por los grupos de desarrollo. Sin embargo, no cuenta con metodologías claras para llevar un control de los módulos que pudieran ser reusables. Por lo que se recomienda hacer un análisis técnico que pueda determinar los posibles módulos que puedan ser utilizados creando un repositorio de módulos reusables para proyectos futuros.

2.3.3.2.8 Documentación en el ciclo de vida, DOCU

Para el caso PS002 fue calificado con un valor de “normal”. Similar a la obtenida en los proyectos del KBC, lo que indica que ciertas etapas se encuentran documentadas. Se debe tomar en cuenta, igual que el caso de estudio anterior, que la responsabilidad de inspeccionar la debida documentación en cada etapa del ciclo de vida del proyecto recae en el LIDER.

2.3.3.2.9 Experiencia en la aplicación, APEX

Este multiplicador mide la experiencia del equipo de desarrollo en la aplicación. El caso PS002 fue calificado con un valor de "normal", lo que significa que la experiencia en desarrollar este tipo de aplicación es aproximadamente 1 año. De acuerdo a KBC, el incremento del esfuerzo es de un 11%.

A pesar de que muchas de las empresas realizan inducción a los nuevos miembros, se recomienda disminuir el impacto en el esfuerzo total a consecuencia de este multiplicador.

2.3.3.2.10 Uso de las herramientas de software, TOOL

Este multiplicador mide el uso de herramientas de software, el cual tiene una calificación "normal" para este caso de estudio.

Aunque la influencia de este multiplicador no es tan alta, según lo especificado por KBC, el valor "más bajo" del esfuerzo máximo se incrementaría es del 6%.

2.3.4 Reportes usables para el futuro

Realizar un análisis por cada multiplicador de esfuerzo, no tiene como objetivo analizar individualmente un proyecto, sino proporcionar una

herramienta que ayude a evaluar la productividad de un departamento de desarrollo dentro de una empresa específica. Debido a que los datos por empresa participante sólo corresponden a un proyecto, es probable que las tendencias no sean del todo precisas, pero a pesar de ello, éstas muestran dónde y cómo se puede mejorar la productividad y, cuando se cuente con más datos, se pueda medir si lo sugerido en realidad fue válido y medible.

2.3.4.1 Influencia de los Multiplicadores de Esfuerzo.

En las secciones 2.3.2.2 y 2.3.3.3 se detalla de forma individual y tratando de proporcionar la mayor cantidad de información para detectar las posibles áreas de mejora de la productividad en cada empresa. En el futuro este análisis puede ser utilizado para comprobar si las aéreas detectadas y las acciones tomadas tuvieron una influencia negativa o positiva en futuros proyectos de la empresa participante.

2.3.4.2 Frecuencia de los Manejadores de Costo.

Una tabla de frecuencia de los multiplicadores de esfuerzo y factores de escala, por cada proyecto participante, indicará las calificaciones más comunes otorgadas a los manejadores de costo. Esto proporciona información acerca de la calificación estándar para los manejadores dentro de la empresa. Sin embargo, existen otros beneficios si se contara con la

participación de más proyectos dentro de una misma empresa, pues adicional a lo mencionado, el gráfico ayudaría a detectar valores atípicos en los manejadores de costo y el punto más importante es que ayudaría a detectar un cambio de calificación en algún manejador específico debido a alguna acción tomada para que la productividad mejore.

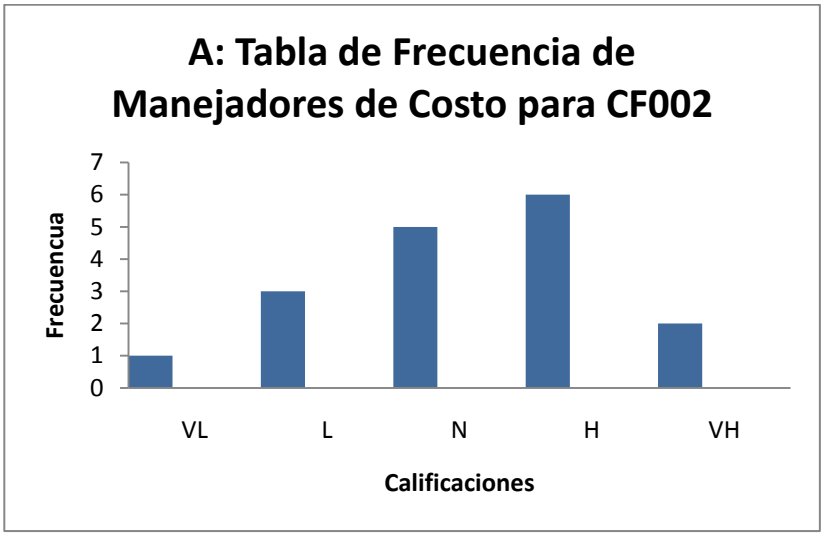


Ilustración 2: Frecuencua de Manejadores de Costo para CF002

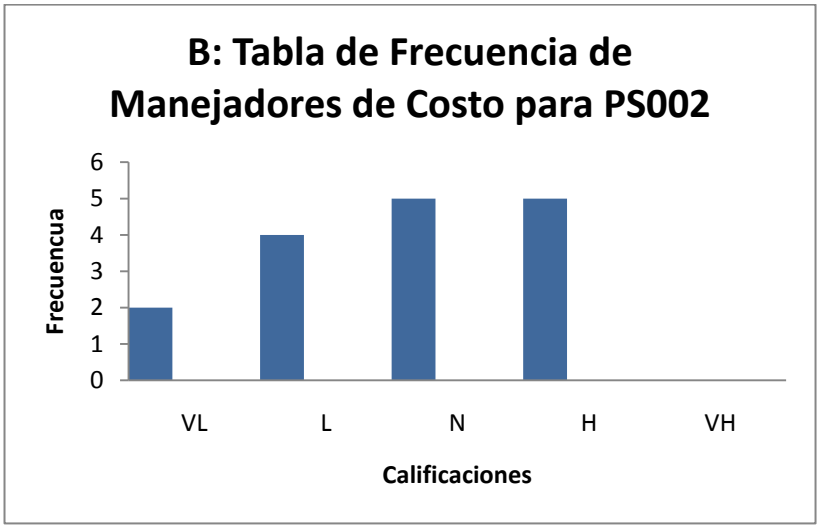


Ilustración 3: Frecuencia de Manejadores de Costo PS002

2.3.4.3 Otros

Adicionales a los reportes ya mencionados, existen 3, que debido al poco volumen de datos no son aplicables para los proyectos en análisis; sin embargo hablaremos un poco de ellos en esta sección:

- Carga de trabajo en relación con el número de líneas de código.

  El objetivo de éste informe es llegar a identificar las tendencias referente a qué tipos de proyectos son menos/más productivos que otros.

- La productividad en función del número de equipos que registran carga de trabajo. Con este informe se busca probar si el hecho de trabajar con múltiples equipos reduce o no la productividad significativamente.

- La productividad en función de un periodo de tiempo.

  Este informe que se construye relacionando el esfuerzo calculado por COCOMO II y el esfuerzo real en el tiempo, ayudará a interpretar la tendencia de las mejoras en la productividad.

CAPITULO 3

3. REFLEXIONES ACERCA DEL PROYECTO

3.1 Puntos importantes y límites de COCOMO II

En esta sección hablaremos de algunos aspectos del uso de COCOMO II además de sus límites como modelo de medición de la productividad en 2 empresas ecuatorianas.

Una de las principales fortalezas de este modelo es su extensa lista de características medibles por proyecto (multiplicadores de esfuerzo y factores de escala), puesto que sólo con los valores de estos manejadores de costos, sin aplicar el modelo, se puede detectar por qué un proyecto es más o menos productivo que otro. Gracias a esta fortaleza, logramos analizar los proyectos y sugerir mejoras para las empresas participantes.

Una vez que se definió claramente la forma en que serían contadas las líneas de código en las empresas participantes, realizar los cálculos para el modelo se convierte en una tarea automática, razón suficiente para calificar a COCOMO II como un modelo rápido y relativamente fácil de aplicar, por lo cual las empresas participantes deberían enfocarse en mejorar el proceso de registro y conteo de líneas de código y formar una base de datos sólida que ayude a realizar análisis basado en experiencias.

Una falencia al aplicar COCOMO II recae en la manera poco objetiva en que los involucrados pueden llenar las hojas de datos ya que esto influye directamente en los resultados del proyecto.

El hecho de que se realice una calibración inicial del método de acuerdo al ambiente local de cada empresa se convierte en un punto a favor al momento

de realizar un análisis interno, pero desemboca en un gran problema si se requiere realizar un análisis entre proyectos de empresas diferentes. Este es un factor crítico determinante del cual dependen los resultados que se desean obtener.

Finalmente, encontrar un punto de equilibrio para resolver las paradojas del conteo de líneas de código se convirtió en la tarea más larga de este proyecto; es importante realizar una buena determinación de los factores de conversión y la manera de sobrellevar la reutilización. Este punto no debe ser descuidado para el futuro, debe ser atendido siempre que haya un cambio en el entorno de medición.

3.2 Parámetros de impacto en la productividad

Las mediciones no solo nos ayudan a comprender y controlar lo que está sucediendo en los proyectos, sino que ayudan a mejorar los procesos y productos.

El paper "Improving software productivity" (19) Boehm identifica las siguientes estrategias para mejorar la productividad: obtener lo mejor de la gente, hacer los diferentes ciclos de desarrollo más eficientes, eliminar fases innecesarias en el desarrollo, eliminar el retrabajo, construir productos simples y componentes reusables. En el modelo COCOMO II (10) de Boehm, los atributos de un proyecto de software que influyen en el esfuerzo necesario para completar un proyecto y por ende en la productividad, son representados por los manejadores de costo. Controlar estos factores nos conducen a un sinnúmero de oportunidades para mejorar la productividad. Adicional a esto, los valores otorgados a estos manejadores de costo dan una estimación

previsible de un posible aumento/disminución de la productividad. Es importante acotar que la mayoría de los multiplicadores de esfuerzo no solo toman en cuenta factores a nivel de proyecto, sino a nivel de la empresa en general.

3.3 Reusar para mejorar la productividad

El Reuso puede mejorar tanto la productividad como la calidad del software.

Aunque el desarrollo de software es un proceso creativo y el producto final entregado hacen que en la mayoría de ocasiones este proceso sea único, existen ciertas partes del proceso o partes de otro producto que puede ser reusadas, factor que puede tener influencia positiva o negativa en la productividad. El porcentaje de incremento de productividad puede llegar al 350% cuando se reusan componentes de calidad muy alta (19).

No es correcto pensar que solo el código puede ser reusable, pues existen componentes como requerimientos, diseño, datos de pruebas e incluso documentación de usuario que también son aptos para reusar (19).

Fenton (1) indica que el re-uso permite al desarrollador concentrarse en nuevos problemas en lugar de continuar resolviendo problemas viejos una y otra vez.

Usar patrones, constituye una excelente vía para mejorar la calidad y la productividad.

CONCLUSIONES

En este proyecto podemos concluir que:

1. El proceso de selección del método con el que se midió el tamaño de software constituye la piedra angular de este proyecto, junto con la elección del modelo de Estimación. El conteo de líneas de código, a pesar de que llevó consigo la resolución de diferentes paradojas, se acopló perfectamente al modelo de estimación de productividad COCOMO II Post-Arquitectura. Esta selección se realizó en base al entorno de las empresas participantes, ya que cada factor que pudo haber influido en el cálculo de la productividad fue tomado en cuenta durante los cálculos respectivos.

2. Es importante recalcar que este proyecto contribuyó con un estudio realizado por la Phd Lotte de Rore en KULEUVEN Katholieke Universiteit Leuven de Bélgica con lo siguiente:
    a. Se tuvo la oportunidad de analizar, en conjunto con la Phd. Lotte de Rore, la problemática del análisis de la productividad desde el punto de vista de empresas desarrolladoras de software ecuatorianas con proyectos de un tamaño menor a los analizados en el caso KBC; situación que con llevo a verificar que la metodología aplicada en Bélgica es válida para emitir un criterio de productividad a nivel local.
    b. A pesar de que los resultados locales fueron productivos, se deja un precedente en el personal capacitado y un repositorio de datos que permitirán a las empresas continuar y escalar el análisis a proyectos de aún mayor tamaño que los actuales.

c. Los datos recolectados en cuanto a los multiplicadores de esfuerzo, permitirán a los analistas expertos, que emitieron su criterio en el caso KBC, formar un criterio del estado de los factores de producto, plataforma, personal y de proyecto en PYMES ecuatorianas.

d. La tabla denominada: "Tabla básica para el cálculo de la productividad" creada por nosotros, después de los ajustes realizados en base a las observaciones de los participantes, se convirtió en una herramienta muy útil para la captura de información en el conteo de líneas de código. Esta tabla puede ser utilizada en empresas que no cuentan con ningún método automático para realizar este procedimiento.

3. Durante el proceso desarrollo de la pre-prueba pudimos observar que el uso de un método de estimación de ambas empresas en estudio era muy bajo. Su personal, entre los que se encontraban egresados de la carrera Computación y una minoría con título de 3er nivel en Ingeniería en Computación, tuvo que recibir dos inducciones sobre el método de estimación y del proceso de medición de la productividad en general. Esto se debe a que los conceptos adquiridos en las materias universitarias no son aplicados por completo, pues se prefiere utilizar la experiencia adquirida que algún método específico, lo que conlleva a un sinnúmero de situaciones que aumentan el esfuerzo y por ende disminuyen la productividad.

4. Otra conclusión importante que nos dejó el proceso de pre-prueba tiene que ver con la resolución de las paradojas para el conteo de líneas de código. Fue

importante que junto con el Jefe de Desarrollo y los Desarrolladores se haya llegado a un conceso por cada paradoja pues de esta forma se garantizó que la medición fuera uniforme.

5. Durante el proceso de cálculo de la productividad de los datos finales, pudimos darnos cuenta, que a pesar de tener una política de reutilización, no se lleva una ficha de detalles de estos módulos y la mayoría del equipo de desarrollo desconoce datos como el número de líneas de código, importantes para el cálculo de la productividad.

6. El análisis realizado en Ecuador es una pequeña porción del proyecto realizado en Bélgica, sin embargo se proporcionó datos importantes los cuales fueron analizados por la Phd. Lotte de Rore. Este análisis arrojó que ambos proyectos locales son productivos comparados con los proyectos analizados en KBC.

   Sin desmerecer el trabajo de las empresas desarrolladoras locales, se piensa que estos resultados se deben a que el tamaño de los proyectos locales es mucho menor a los proyectos analizados en el caso KBC en Bélgica. Nuestras empresas cuentan con métodos empíricos para lograr una productividad satisfactoria pero es necesario que tomen en cuenta ciertos aspectos que el análisis de los multiplicadores de esfuerzo arrojó durante el proceso de selección del modelo de estimación que evaluó la aproximación tamaño/esfuerzo.

RECOMENDACIONES

Con la experiencia adquirida en este proyecto podemos hacer las siguientes recomendaciones:

1. Para estudios futuros, se recomienda tomar en cuenta el nivel profesional de los desarrolladores en las empresas participantes, debido a que esto puede marcar una variación en el número de líneas de código (tamaño) que éstos utilicen e influir directamente en la productividad de un proyecto.

2. Otro punto importante que se debe tomar en cuenta que se podría implementar algún método adicional que permita no solo utilizar el criterio experto del Jefe de Desarrollo o Lider del Equipo, otorgar una calificación a un multiplicador de esfuerzo.

3. En el análisis del multiplicador de esfuerzo LTEX (experiencia en el lenguaje y herramienta) para el proyecto CF002, se detectó que la empresa participante debe esforzarse por aumentar este valor. Se recomienda analizar la forma en la que se componen los grupos de desarrollo, puesto que algunos pueden ser considerados débiles y otros fuertes por la experiencia que poseen.

4. En el análisis del multiplicador de esfuerzo DATA (tamaño de la base de datos) del proyecto CF002, detectamos que a pesar de que hay una influencia mínima en el esfuerzo total, existe la tendencia a que este valor se eleve dentro de la empresa, razón por la cual recomendamos planificar

políticas sobre la cantidad de datos requeridos para realizar pruebas. Adicionalmente, deben de asignar a una persona en particular la responsabilidad de generar y administrar éstos.

5. En el análisis de los multiplicadores de esfuerzo STOR (restricción del almacenamiento principal) y TIME (restricción en el tiempo de ejecución) del proyecto CF002, detectamos que al igual que para el multiplicador DATA es necesario planificar qué medidas tomar en caso de que los valores se eleven, ya que esto implicaría mayor esfuerzo en etapas ajenas al desarrollo, particularmente en el momento en que la arquitectura está siendo definida. Lo que se busca es disminuir la probabilidad de que se presenten problemas en la etapa de desarrollo que luego impliquen un retroceso a etapas iniciales y por ende el esfuerzo se incremente significativamente.

6. En el análisis de los multiplicadores de esfuerzo RUSE (desarrollo para reusabilidad) del proyecto CF002, detectamos un valor “alto”, pero a pesar de ello, recomendamos pulir el proceso de reutilización y disminuir la variable AA, la cual, representa el esfuerzo necesario para decidir qué módulo disponible puede ser reutilizado en una determinada aplicación.

7. Definitivamente ambas empresas deben tomar muy en cuenta que la falta de documentación es un punto débil. Los manuales de usuario no es lo único que se debe documentar sino todo aquello que les permita crear una base de conocimiento sólida para mejorar continuamente su proceso de desarrollo

8. Para el multiplicador DATA (tamaño de base datos) del caso de estudio PS002, recomendamos planificar políticas para reducir el esfuerzo en generar y administrar la base de datos para realizar las pruebas. Los miembros designados en realizar las pruebas deben ser agentes externos al equipo de desarrollo para garantizar la efectividad de las mismas.

APENDICES

APÉNDICE A: PRE-TEST

A1: Documento Final de Inducción

1. *Introducción*

El presente documento, detalla las actividades a realizarse en el proceso de Pre-Pruebas de las Tablas de Datos que serán utilizadas, para calcular la productividad en los proyectos de software participantes, durante el desarrollo del proyecto de graduación: Medición de la productividad de proyectos de Software en dos empresas locales.

Este proceso tiene una duración de 2 semanas, durante éste tiempo se tomarán datos en 3 tablas diferentes: Factores de Escala, Multiplicadores de Esfuerzo, y Tabla Básica para la productividad. [Ver Anexos]. De acuerdo al tipo de tabla, se necesitará la colaboración de los diferentes miembros del departamento de desarrollo como: Jefe de Grupo de Desarrollo, Desarrollador, Jefe de Departamento de Desarrollo, entre otros. EL participante no deberá tomar tiempo adicional para realizar ésta prueba, es importante que los datos en las tablas se llenen de manera simultánea a la realización de las tareas de desarrollo.

2. *Objetivos*
   a. Objetivo General.- Validar las tablas: Factores de Escala, Multiplicadores de Esfuerzo y Tabla básica para la productividad con la finalidad de que se constituyan en herramientas para el cálculo de productividad con el modelo de estimación COCOMO,

independientemente del tipo de proyecto de software y de la empresa o grupo que lo desarrolle.

b. Objetivos Específicos:
    i. Preparar y Familiarizar a los Participantes involucrados, con el procedimiento que se llevará a cabo en el proyecto.
    ii. Tomar en cuenta las debilidades de la Pre-Prueba de manera que sirvan como retroalimentación para mejorar el procedimiento de toma de datos.
    iii. Formar un criterio preliminar con los resultados obtenidos en la Pre-Pruebas.
    iv. Tomar medidas correctivas en las tablas de Datos, de ser necesario, de tal forma que no se presenten imprevistos al momento de realizar la toma oficial de datos.
    v. Comparar, a pequeña escala, los resultados obtenidos en la Pre-Prueba con los resultados finales, esto para definir si existe alguna relación entre los resultados obtenidos.
    vi. Definir puntos críticos durante la toma de datos, para prevenir inconvenientes en la toma de datos oficial.

3. Consideraciones

Durante el desarrollo de la Pre-Prueba se deben tomar en cuenta las siguientes consideraciones:

- La tabla debe llenarse en el momento en que se esté realizando la tarea de desarrollo, de lo contrario, los datos podrían variar si el tiempo transcurrido es muy amplio existe la posibilidad que los datos ingresados en las tablas de datos tengan un porcentaje de error elevado.
- Las respuestas deben ser lo más aproximadas a la realidad de la empresa y/o proyecto, todas las preguntas consultadas en las tablas influyen directamente en el cálculo de la productividad.
- En caso de existir alguna duda al momento de llenar una casilla en la tabla de datos, se debe recurrir a la persona encargada de dirigir la prueba en su respectiva empresa o consultar a los anexos de éste documento, en donde se explica detalladamente cada campo de las tablas.

4. *Participantes*

Por parte del grupo desarrollador del proyecto:

- Manuel Olvera Alejandro: Encargado de la empresa Palosanto Solutions.
- Lohana Lema: Encargada de la empresa Dayscript.

En Cada empresa participante se solicitará la participación de:

- Jefe de Departamento de Desarrollo.
- Jefe de cada Equipo de desarrollo participante.
- Cada Desarrollador que intervenga en el proyecto escogido para el análisis.

5. *Anexos.*

a. *Tabla 1: Factores de Escala*

- Esta Tabla debe llenarse por CADA PROYECTO a analizar.
- Escala para llenar la tabla:

1	2	3	4	5	6
Muy Bajo	Bajo	Normal	Alto	Muy alto	Extra Alto

- *Tabla de Datos*

1.1: Factores de Escala - por cada proyecto -		
Factor	**Breve Explicación**	**Calificación**
Precedentes	¿El proyecto que se analizará es similar a otros que se han realizado antes?	
Flexibilidad de Desarrollo	¿El proyecto es Flexible respecto a sus requerimientos?	
Resolución de Riesgos y Arquitectura	¿Se ha tomado mucha atención a la arquitectura? ¿Se han tomado en cuenta los riesgos del proyecto?	

Cohesión del Equipo	¿Hay problemas de sincronización entre stake-holders?	
Maduración del Proceso	¿Cuál es el nivel CMMI del equipo de desarrollo?	

- *Explicación de los Campos:*

Precedentes: Es usado para conocer si existieron previamente proyectos similares, en base al modelo del negocio o de las características solicitadas por el cliente.

Flexibilidad: Mide que tanto el modelo usado se adapta a los nuevos o cambios de requisitos. Resolución de riesgos y arquitectura, para el desarrollo del proyecto si tiene procesos de medición del riesgo y que tanto considera dentro del desarrollo.

Resolución de Riesgos y Arquitectura: Se tiene en cuenta las medidas que se tomarán para la eliminación de Riesgos.

Cohesión del equipo: Se refiere a la comunicación que existe entre los actores involucrados en el desarrollo, desde el cliente hasta el desarrollador, pasando por los diferentes coordinadores, usuarios potenciales y demás interesados.

Maduración del Proceso: Determina si la empresa tiene alguna certificación o en su caso el modelo de procesos para desarrollar Software es similar al que propone CMMI en alguno de sus cinco niveles: Marcar Muy Bajo si no se tiene ningún nivel, marcar Bajo si se tiene el Nivel 1 CMMI, Marcar Normal si se encuentra en Nivel 2,

Marcar Alto si se encuentra en Nivel 3, Marcar Muy Alto si se encuentra en Nivel 4 y Marcar Extra Alto si se encuentra en Nivel 5.

b. *Tabla 2: Multiplicadores de Esfuerzo.*

- Esta tabla debe llenarse por cada módulo del proyecto a Analizar.
- Escala para llenar la tabla:

1	2	3	4	5	6
Muy Bajo	Bajo	Normal	Alto	Muy alto	Extra Alto

2: Multiplicadores de Esfuerzo en la Post-Arquitectura del Modelo COCOMO II – Por cada módulo del proyecto.		
Multiplicador	**Breve Explicación**	**Calificación**
1: Fiabilidad del Software	¿Qué tan grande es el efecto de un fallo de Software?	
2: Tamaño de la Base de Datos	¿Qué cantidad de datos de prueba se necesitarán alojar en la base de datos?	
3: Complejidad del Producto	¿Cuán complejo es el producto con respecto al control computacional, operaciones que dependen de dispositivos y opresiones de administración de Interfaz de usuario?	
4: Reusabilidad en el Desarrollo	¿Los Componentes a desarrollar serán reutilizables?	
5: Documentación	¿Cuántas etapas del ciclo de vida de desarrollo están documentadas?	
6: Restricciones en el tiempo de	¿Existe alguna exigencia por parte del cliente en	

Ejecución	los tiempos de respuesta en tiempo de ejecución?	
7: Restricciones en la Base de Datos Principal	¿Existe algún tipo de restricción en cuanto al porcentaje de uso de la base de datos principal?	
8: Volatibilidad de la Plataforma	¿Hay grandes y frecuentes cambios en lo que a plataforma se refiere?	
9: Capacidad de Análisis	¿Cuál es la capacidad de los Analistas?	
10: Capacidad del Programador	¿Cuál es la capacidad de los programadores?	
11: Continuidad del Personal	¿Cuál es la frecuencia anual de rotación de Personal?	
12: Experiencia en las Aplicaciones	¿Cuál es la experiencia del equipo que participa en el proyecto en el desarrollo de éste tipo de aplicaciones?	
13: Experiencia en la Plataforma	¿Cuál es la experiencia de equipo que participa en el proyecto en la plataforma a utilizar?	
14: Experiencia en el lenguaje de programación y Herramientas	¿Cuál es la experiencia del equipo que participa en el proyecto en el uso del lenguaje de programación y herramientas a emplear?	
15: Uso de las Herramientas de Software	¿Se utilizó una herramienta de software existente para desarrollar el producto?	
16: Desarrollo Multisite	¿Existe un soporte de comunicación disponible?	
17: Planificación del Desarrollo	¿Existe calendario de restricciones impuesto sobre el equipo del	

	proyecto?	

- *Explicación de los Campos:*

1: Calificar MUY BAJO si el efecto del fallo de Software únicamente es producir pequeños inconvenientes, si el fallo llegase arriesgar una vida humana o situaciones semejantes, entonces calificar EXTRA ALTO.
2: Esta medida pretende capturar los efectos que tienen requerimientos de gran cantidad de datos sobre el desarrollo del producto. Calificar 1 si no se necesitan o se necesitan pocos datos de prueba. Calificar 6 si el volumen de datos debe ser muy grande
3: Consultar la Tabla 2.a que detalla la información necesaria.
4: Este parámetro de coste mide el esfuerzo adicional necesario para la construcción de componentes que puedan reutilizarse en el actual o en futuros proyectos. Este esfuerzo es consumido durante la creación de un diseño software más genérico, una documentación más elaborada, y un chequeo más exhaustivo para asegurarse de que los componentes quedan listos para ser utilizados en otras aplicaciones. Calificar 1 o 2 en caso de no realizar ninguna forma de reutilización. Calificar 3 Si se existe ésta característica solo en el programa, Calificar 4 si existe ésta característica en todo el proyecto, Calificar 5 si existe **ésta** característica en la línea del producto, Calificar 5 si existe ésta

característica en las múltiples líneas de productos que maneje la empresa.
5: Es evaluada en términos de la adecuada documentación del proyecto en el ciclo de vida. Esta escala va desde Muy Bajo (se observan algunas necesidades en el ciclo de vida) hasta Muy Alto (muchas y grandes necesidades del ciclo de vida).
6: Esta es una medida de la restricción del tiempo de ejecución impuesta en un sistema software. Las medidas se expresan en términos de porcentaje de tiempo de ejecución disponible que se espera que sea usado por el subsistema ó sistema que consume el recurso de tiempo de ejecución.
7: Esta medida representa el grado de restricción de almacenamiento principal impuesto a un sistema ó subsistema software. Los valores van desde nominal, menos que el 50%, a Extra Alto, 95%.
8: Plataforma comprende el conjunto formado de Software y Hardware que se utilizará durante el desarrollo del proyecto. La plataforma incluye cualquier compilador o ensamblador utilizado para desarrollar el sistema. Este rango va desde Bajo, donde existen cambios importantes en la plataforma cada 12 meses, hasta Muy Alta, con cambios importantes cada dos semanas aproximadamente.
9: Analizar y calificar los siguientes atributos que deberían de ser

considerados entre los miembros del equipo: la habilidad, eficiencia, habilidad de comunicación y cooperación.
10: La Capacidad del programador se debe medir en base a la habilidad o capacidad de los desarrolladores para compaginar con su grupo mas no a su individualidad.
11: Se debe medir en términos del personal que retorna anualmente al proyecto: desde 3%: Muy Alto, al 48%, Muy Bajo.
12: Esta valoración es dependiente del nivel de experiencia que se posee en las aplicaciones por el equipo de desarrollo. Un rango de Muy Bajo para una experiencia menor de 2 meses, y Muy Alto si se posee una experiencia igual o superior a 6 años.
13: Se valora de la misma manera que la experiencia en Aplicaciones.
14: El desarrollo del software incluye el uso de herramientas para realizar los requerimientos, y el análisis y diseño de la representación, manejo de la configuración, extracción de la documentación, gestión de librerías, formateo y estilo de programa, chequeo de consistencia y adicionalmente a la experiencia en programación con un lenguaje específico. Un rango de Muy Bajo para una experiencia menor de 2 meses, y Muy Alto si se posee una experiencia igual o superior a 6 años.

15: Valorar este punto tomando en cuenta las características de las herramientas de Software que van desde las simples herramientas de edición y codificación, a las que se les asigna un valor asociado de Muy Bajo, hasta las herramientas integradas de gestión del ciclo de vida, asignándoles un valor de Muy Alto.
16: Para éste Multiplicador, se debe de tomar en cuenta 2 características: Desarrollo Multisite por la Ubicación, va desde Internacional (muy bajo), Multiciudad y multicompañía, Multiciudad o Multicompañia (normal), Misma ciudad, Mismo edificio, Misma situación (muy alto); y el Desarrollo Multisite por comunicaciones, va desde: Teléfono y/o correo postal (muy bajo), Teléfono Individual y/o Fax, Email, Banda Ancha, Videoconferencia o cualquier comunicación electrónica de banda Ancha y Multimedia Interactiva (muy alto). Un promedio de estas dos características dará valor a éste punto.
17: Este Multiplicador se mide en términos del porcentaje de planificación que se alarga o acelera, planificaciones temporales aceleradas tienden a producir más esfuerzo en las últimas fases de desarrollo debido a que más asuntos son dejados para ser determinados posteriormente. Una planificación comprimida un 74% es valorada como un porcentaje Muy Bajo, y un alargamiento del 160% es valorada como un porcentaje Muy Alto.

Tabla 2.a: Valoración de la Complejidad del producto según el tipo de módulo:

	Muy baja	Baja	Nominal	Alta	Muy alta	Extra alta
Operaciones de control	Lineas simples de código con poco codigo y sin estructuras anidadas: DOs, CASEs, IFTHENELSs. Llamadas a procedimientos simples.	Uso anidado de operadores de programación estructurada de forma directa. Mayoritariame nte uso de predicados simples.	Mayoría de los anidamientos de operadores son simples. Algún control entre modulos. Tablas de decisión. Paso de mensajes, incluyendo proceso distribuido a medio nivel	Programación altamente anidada con mucha composición de predicados. Procesos distribuidos. Control en tiempo real simple.	Codigo reentrante y recursivo. Manejo de interrupciones con prioridad. Sincronización de tareas, control en tiempo real complejo.	Múltiple planificación de recursos con prioridades que cambian dinamicamente . Control a nivel de microcódigo. Control distribuido en tiempo real.
Operaciones de computo	Evaluación simple de expresiones: A=B+C*(D-E)	Evaluación de expresiones a nivel medio: SQRT(B^2-4*A*C)	Uso de rutinas estandar, matemáticas y estadísticas. Operaciones básicas sobre matrices o vectores.	Analisis numérico básico: multivariable, interpolacion, ecuaciones diferenciales de primer orden.	Dificil y estructurado análisis numérico: ecuaciones matriciales, ecuaciones diferenciales parciales. Paralelización simple.	Analisis numérico dificil y no estructurado: análisis altamente preciso de ruido, datos estocásticos. Paralelización compleja.
Operaciones dependientes de dispositivos	Lecturas simples, escritura de declaraciones con formas simples.	No es necesario conocimiento sobre procesadores I/O particulares. La I/O se realiza a niivel de GET/PUT.	El proceso de I/O incluye selección de dispositivos, comprobación de estado, y proceso de los errores.	Operaciones de I/O a nivel fisico (traducción de direcciones fisicas, posicionamient os en memoria, lecturas, ...).	Rutinas para diagnóstico, de servicio, de enmascaramie nto de interrupciones. Manejo de lina de comunicacione s. Sistemas de proceso intensivo.	Codificación con control de tiempos de dispositivos, operaciones de microprograma ción. Sistemas de proceso crítico.
Operaciones de manejo de datos	Arrays sencillos en memoria principal. Consultas y actualizaciones simples de la base de datos.	Tramtamiento de ficheros de forma simple, sin cambios en la estrucutra de datos, sin ficheros intermedios, ... Consultas y actualizaciones moderadas de la base de datos.	Entrada desde varios ficheros, y salida única. Cambios simples en la estructura. Consultas y actualizaciones complejas.	Simples señales de activación flujos de datos. Reestructuraci ón de datos compleja.	Coordinación de bases de datos distribuidas. Señales de activación complejas. Optimización de búsquedas.	Altamente acoplados, relacionados dinámicamente y con estructura de objetos. Manejo de lenguaje natural de datos.
Operaciones para gestión del interfaz de usuario	Formularios de entrada simples. Generador de listados.	Uso de GUIs simples.	Uso simple de elementos de interfaz.	Desarrollo de elementos de interfaz. Multimedia y I/O simple con voz.	Moderadament e complejo, 2D/3D, gráficos dinámicos.	Multimedia compleja, realidad virtual.

c. *Tabla 3: Tabla Básica para el cálculo de la productividad.*

Nombre de la Empresa:	
Nombre del Proyecto:	
Nombre del módulo:	
Número de Equipos que está trabajando en el proyecto:	
Equipo de Desarrollo:	
Nombre del Desarrollador:	
% de familiaridad del desarrollador con el software existente:	
Código Único:	

1: # líneas de código en la tarea inicial

2: # líneas de código generado automáticamente

3: % diseño modificado

4: # líneas nuevas del lado del servidor

5: # líneas nuevas del lado del cliente

6: # líneas modificadas del lado del servidor

7: # líneas modificadas del lado del cliente

8: tiempo invertido en la tarea

Tarea	**(1)**	**(2)**	**(3)**	**(4)**	**(5)**	**(6)**	**(7)**	**(8)**

- *Explicación de los Campos:*

  El campo 'Tareas' sirve para detallar cada una de las acciones relacionadas con el desarrollo.

  El campo '#Líneas de código en la tarea inicial' se llena cuando la tarea es un nuevo desarrollo, considerado como una nueva acción.

  El campo '#Líneas de código generado automáticamente' se llena solo si el código fue generado por alguna herramienta.

  El campo '% diseño modificado' corresponde a el porcentaje que el diseño original haya cambiado con respecto al original, para cuando la tarea es una modificación.

  El campo '# líneas nuevas del lado del servidor', se ingresa la cantidad de líneas que se desarrollaron para ser ejecutadas de lado del servidor cuando en un incremento.

  El campo '# líneas nuevas del lado del cliente' corresponde a la cantidad de líneas desarrolladas para ser ejecutadas en los navegadores, para cuando son códigos de incrementos.

  El campo '# líneas modificadas del lado del servidor' se ingresan la cantidad de líneas que fueron modificadas, ya sean por cambios de requisitos o por corrección de errores.

  El campo '# líneas modificas del lado del cliente" se registra la cantidad de líneas de código que se ejecuta en el navegador que han sido modificadas.

El campo 'tiempo invertido en la tarea' corresponde al tiempo total efectivo usado para desarrollar la tarea, este campo es obligatorio.

d. *Tabla 4: Tabla de Datos para el control de módulos reusados.*

Control de Módulos Reusados				
		Código único del Proyecto		
Nombre del Módulo o Componente Reutilizado	**Líneas estáticas del módulo o Componente**	**Líneas Adaptadas Cliente**	**Líneas Adaptadas Servidor**	**Porcentaje de esfuerzo para la adaptación**

- *Explicación de los Campos:*

El campo 'Nombre del módulo o Componente Reutilizado' sirve para detallar el nombre que se otorgó al módulo que se reutilizó en éste proyecto.

El campo 'Líneas estáticas del módulo o Componente' se llena con el número de líneas que el módulo o componente a ser reutilizado tiene, si el componente no ha sufrido ninguna actualización, este número no debe cambiar con el tiempo.

El campo 'Líneas adaptadas Cliente' representa el número de líneas que el desarrollador tuvo que escribir del lado del cliente para poder adaptar el módulo reutilizado al proyecto que se está analizando.

El campo ‘Líneas adaptadas Servidor’ representa el número de líneas que el desarrollador tuvo que escribir del lado del servidor para poder adaptar el módulo reutilizado al proyecto que se está analizando.

El campo ‘porcentaje de esfuerzo para la adaptación’, representa en porcentaje, el esfuerzo que dedicó el desarrollador para habilitar el módulo o componente reusable en el proyecto en análisis.

A2: Encuesta Realizada a los Participantes

Nombre de la Empresa:	
Nombre del Proyecto:	
Nombre del Participante	
Fecha:	
Cargo:	

Esta Encuesta tiene como finalidad, medir el grado de preparación que los participantes tuvieron al momento de realizar el Pre-Test. Es importante que sus respuestas sean 100% objetivas, ya que nos servirán como retroalimentación para realizar medidas correctivas en el proceso oficial de toma de datos.

1. ***Cuál de las siguientes opciones fueron explicadas en la inducción al Pre-Test.***
    a. Marco General del Pre-Test (de que se trataba, porque se lo realiza):
    b. Tabla 1: Para qué sirve y que significa cada campo a llenar:
    c. Tabla 2: Para qué sirve y que significa cada campo a llenar:
    d. Tabla 3: Para qué sirve y que significa cada campo a llenar:

2. ***Leyó todo el documento de Planificación antes de empezar a realizar el Pre-Test?***

3. ***¿Consulté las dudas a la persona encargada del Pre-Test?***
    ¿Cuántas consultas aproximadamente?

4. ***En qué nivel del 1 al 5 el Documento Plan de Desarrollo de Pre-Test fue útil para la realización de la toma de datos.***

    1: No lo utilicé.

    2: Lo utilicé poco.

    3: Lo utilicé algunas veces.

    4: Lo utilicé la mayoría del tiempo.

    5: Siempre lo recurría al documento cuando tenía una duda.

5. ***Califique del 1 al 5, siendo 1 el valor más bajo y 5 el valor más alto, la claridad con que se explica cada Factor de Escala en la columna de "Breve explicación" y en la sección "Explicación de Campos" que se encuentra al final de la Tabla 1.***

    1: No se entiende la Breve explicación, no me ayuda a comprender que debo calificar.

    2: La explicación es muy simple, necesita más detalle.

    3: La explicación es simple, pero se entiende.

    4: La explicación está clara y correcta, pero aún tengo un poco de dudas.

    5: La explicación es muy clara y pude calificar sin tener grandes dudas.

6. ***Califique del 1 al 5, siendo 1 el valor más bajo y 5 el valor más alto, la claridad con que se explica cada multiplicador de esfuerzo en la columna de "Breve explicación" y en la sección "Explicación de Campos" que se encuentra al final de la Tabla 2.***

    1: No se entiende la Breve explicación, no me ayuda a comprender que debo calificar.

    2: La explicación es muy simple, necesita más detalle.

3: La explicación es simple, pero se entiende.

4: La explicación está clara y correcta, pero aún tengo un poco de dudas.

5: La explicación es muy clara y pude calificar sin tener grandes dudas.

7. ***Con Respecto a la tabla #3, por favor explicar bajo que criterio usted llenó los siguientes campos:***
    a. # líneas de código en la tarea inicial.
    b. # líneas de código generado automáticamente.
    c. % de diseño modificado.
    d. # líneas nuevas del lado del Servidor.
    e. # líneas nuevas del lado del Cliente.
    f. # líneas modificadas del lado del Servidor.
    g. # líneas modificadas del lado del Cliente.

8. ***¿Qué porcentaje considera usted, que tiene de familiaridad con las tablas de datos una vez que utilizó en el Pre-Test?***

9. ***¿Se sentiría más seguro si se realizara una nueva inducción antes de la toma de datos oficial?***

10. ***Por favor, utilizar este espacio para alguna observación adicional que no haya sido tomada en cuenta en alguna de las preguntas anteriores.***

A3: Resultados de la Encuesta.

Resultado de las Encuestas a Participantes

#Pregunta	Desarrollador 1 PS		Desarrollador 2 PS		Desarrollador 1 CF	
1	A		A		TODAS	
2	OK		OK		OK	
3	OK	3	OK	N/E	OK	4
4	4		4		4	
5	4		4		4	
6	5		3		3	
7	A	c	a	c	A	C
8	40%		55%		50%	
9	Si		Si		Si	
10	Ninguna		Ninguna		Ninguna	

A4: Datos Tomados.

Factores de Escala:

	Calificación	
	CF001	PS001
Precedentes	2	6
Flexibilidad de Desarrollo	3	5
Resolución de Riesgos y Arquitectura	1	2
Cohesión del Equipo	3	4
Maduración del Proceso	0	1

Multiplicadores de Esfuerzo:

	Calificación	
Multiplicador	CF001	PS001
1: Fiabilidad del Software	3	6
2: Tamaño de la Base de Datos	3	3
3: Complejidad del Producto	6	5
4: Reusabilidad en el Desarrollo	5	6
5: Documentación	1	6
6: Restricciones en el tiempo de Ejecución	3	3
7: Restricciones en la Base de Datos Principal	2	3
8: Volatibilidad de la Plataforma	1	2
9: Capacidad de Análisis	4	5
10: Capacidad del Programador	6	5
11: Continuidad del Personal	4	2
12: Experiencia en las Aplicaciones	4	3
13: Experiencia en la Plataforma	5	3
14: Experiencia en el lenguaje de programación y Herramientas	5	3
15: Uso de las Herramientas de Software	6	3
16: Desarrollo Multisite	3	6
17: Planificación del Desarrollo	5	3

A8: Cálculos y Resultados

Cálculos de Productividad CF001

Cálculo de M.E.	
RELY	1
DATA	1
CPLX	1,74
RUSE	1,15
DOCU	0,81
TIME	1
STOR	1
PVOL	1
ACAP	0,85
PCAP	0,76
PCON	0,9
AEXP	0,88
PEXP	0,85
LTEX	0,84
TOOL	0,78
SITE	1
SCED	1
0,46	2,94
1,36	

Cálculo de F. E.	
PREC	4,96
FLEX	3,04
RESL	7,07
TEAM	3,29
PMAT	7,8
26,16	1,1716
0,91	

Cálculo de Esfuerzo	
AT	93,95
DM	0
CM	98,35
IM	2,5
SU	30
AA	4
UNFM	2
AFF	30,25
AAM	0,32
KSLOC	1,72
Esfuerzo	2,68

Cálculos de Productividad PS001

Cálculo de M.E.

RELY	1,26
DATA	1
CPLX	1,17
RUSE	1,15
DOCU	1,23
TIME	1
STOR	1
PVOL	0,87
ACAP	0,85
PCAP	0,88
PCON	1
AEXP	1
PEXP	1
LTEX	1,12
TOOL	1
SITE	0,86
SCED	1

1,31 2,94

3,84

Cálculo de F. E.

PREC	3,72
FLEX	4,05
RESL	5,65
TEAM	2,19
PMAT	1,56

17,17 1,0817

0,91

Cálculo de Esfuerzo

AT	0
DM	45,38
CM	6,78
IM	0
SU	30
AA	0
UNFM	2
AFF	20,18
AAM	0,1798
KSLOC	4,31
Esfuerzo	1,08

APÉNDICE B: ANÁLISIS FINAL

B1: Datos Tomados.

Factores de Escala	Calificación CF002	Calificación PS002
Precedentes	2	3
Flexibilidad de Desarrollo	3	2
Resolución de Riesgos y Arquitectura	3	2
Cohesión del Equipo	4	3
Maduración del Proceso	1	1

Multiplicadores de Esfuerzo	Calificación CF002	Calificación PS002
1: Fiabilidad del Software	1	1
2: Tamaño de la Base de Datos	3	5
3: Complejidad del Producto	4	3
4: Reusabilidad en el Desarrollo	4	2
5: Documentación	2	3
6: Restricciones en el tiempo de Ejecución	1	3
7: Restricciones en la Base de Datos Principal	1	5
8: Volatibilidad de la Plataforma	1	2
9: Capacidad de Análisis	4	5
10: Capacidad del Programador	5	2
11: Continuidad del Personal	4	2
12: Experiencia en las Aplicaciones	4	3
13: Experiencia en la Plataforma	3	5
14: Experiencia en el lenguaje de programación y Herramientas	5	5
15: Uso de las Herramientas de Software	4	3
16: Desarrollo Multisite	2	1
17: Planificación del Desarrollo	2	3

B2: Cálculos y Resultados.

Cálculos de Productividad CF002

Cálculo de M.E.

RELY	0,82
DATA	1
CPLX	1,74
RUSE	1,07
DOCU	0,91
TIME	1
STOR	1
PVOL	1
ACAP	0,85
PCAP	0,76
PCON	0,9
AEXP	0,88
PEXP	1
LTEX	0,91
TOOL	0,9
SITE	1,09
SCED	1,14

0,72 2,94

2,13

Cálculo de F. E.

PREC	4,96
FLEX	3,04
RESL	4,24
TEAM	2,19
PMAT	7,8

22,23 1,1323

0,91

Cálculo de Esfuerzo

AT	90,95
DM	0
CM	80,05
IM	1
SU	10
AA	2
UNFM	2
AFF	24,31
AAM	0,46
KSLOC	3,05
Esfuerzo	7,1

Cálculos de Productividad PS002

Cálculo de M.E.

RELY	0,82
DATA	1,14
CPLX	1
RUSE	0,95
DOCU	1
TIME	1
STOR	1,05
PVOL	0,87
ACAP	0,85
PCAP	1,15
PCON	1,12
AEXP	1
PEXP	0,91
LTEX	0,91
TOOL	1
SITE	1,22
SCED	1

0,90 2,94

2,64

Cálculo de F. E.

PREC	3,72
FLEX	4,05
RESL	5,65
TEAM	3,29
PMAT	7,8

24,51 1,1551

0,91

Cálculo de Esfuerzo

AT	0
DM	0,27
CM	0
IM	0,0003
SU	0,3
AA	1
UNFM	1
AFF	0,108
AAM	0,011
KSLOC	2,01
Esfuerzo	5,92

Bibliografía

[1]. **N. Fenton and S. Pfleegcr,** *Softsoare metrics: a rigorous and practical approach.* Boston, MA, USA : PWS Publishing Co, Junio 2009.

[2]. . **R. Prcmraj, M. Shepperd, B. Kitchenharn, and P. Forselius.** *An Empirical Analysis of Software Productivity over time. In software Metrics* 11th IEEE International Symposium. págs. 37 -37. Junio 2009

[3]. **Jones, C.** *Programming Productivity.* New York : McG raw-Hill, Agosto 2009

[4]. **IFPUG.** *http://www.ifpug.org,* Agosto 2009.

[5]. **ISO/IEC.** 2092G:2003(E)Software engineering-IFPUG 4.1 Unadjusted functional size measurement method - Counting practices manual, Agosto 2009

[6]. **Cosmic FFP**. *http://ww.cosmicon.com,* Agosto 2009.

[7]. **ISO/IEC.** 19761:2003(E) Software engineering - COSMIC-FFP - A functional size measurement method, Agosto 2009.

[8]. **COSMIC FFP**. History of functional size measurement. *http://www.cosmicon com/historycs.asp,* Septiembre 2009.

[9]. **L. Santillo and H. van Heeringen.** *Proposals for increasing benchmarking data quality of projects measured in cosmic,* Septiembre 2009.

[10]. **B. Boehm, B. Steece, and R. Madachy.** *Software Cost Estimation with Cocomo II with Cdrom.* NJ, USA : Prentice Hall PTR Upper Saddle River, 2000. Septiembre 2009

[11]. **Boehm, B.** *Software Engineering Economics.* NY, USA : Prentice Hall PTR Upper Saddle River, 1981. Septiembre 2009.

[12]. **R. Selby.** *Empirical Analyzing Software Reuse in a Production.* 1988. pp. 176 - 189. Septiembre 2009.

[13]. **B. Clark, S. Devnani-Chulani, and B. Boehm.** *Calibrating the COCOMO II Post Arquitecture Model. Proceedings ofthe 20th international conference on Softuiare engineering.* 1998. pp. 477 - 480. Octubre 2009.

[14]. **S. Chulani, B. Boehm, and B. Steecc.** *Bayesian analysis of empirical software engineering cost models. Software Engineering, IEEE Transactions.* 1999. pp. 473 - 483.

[15]. **Park, R.** *Software Size Measurement: A Frameworkfor Counting.* s.l. : Carnegie Mellon University, Software Engineering Institute, 1992 Octubre 2009.

[16]. **Rollo., A. L.** *Functional size measurement and cocomo -a synergistic approach.*2006. Octubre 2009.

[17]. **T. Harbich and K. Alisch.** *Accuracy of Estimation Models with Discrete.* 2007. pp. 313 - 324. Noviembre 2009.

[18]. **C.Jones.** *Programming productivity.* New York : McGraw-Hill, 1985. Noviembre 2009.

[19]. **B. Clark, S. Devnani-Chulani, and B. Boehm.** *Calibrating the COCOMO II post-architecture model.* Proceedings ofthe 20th international conference on Software engineering, Noviembre 2009.