Desarrollo de una nueva metodología derivada del COCOMO II agregando pautas de calidad.

Obdulia Lorena Franco Araujo
Facultad Politécnica, Universidad Nacional del Este.
Ciudad del Este, Paraguay.
lorenaf86@gmail.com

Resumen

El objetivo de este trabajo es ofrecer una metodología de estimación de costos basada en la metodología COCOMO II que sea aplicable a proyectos de desarrollo de software que utilizan métodos ágiles de desarrollo. El mismo habrá de aportar mejoras en la estimación de costos de proyectos ágiles considerando la productividad del equipo y ofreciendo técnicas flexibles en el desarrollo de software. Se espera establecer un modelo práctico de estimación de costos que incorpore características de proyectos ágiles.
Descriptores: COCOMO II, metodologías ágiles de software.

Abstract

The aim of this work is to provide a cost estimation methodology based on COCOMO II, applicable to software development project using agile development methods. It is expected to bring improvements in the estimation of agile project costs considering team productivity and providing flexible development techniques. A practical model for cost estimation is expected to establish, incorporating agile project features.
Keywords: COCOMO II, agile development software.

1. Introducción.

La creciente complejidad de los desarrollos software que provocó la denominada "crisis del software" se ha tratado de abordar mediante el planteamiento de nuevos métodos, metodologías, técnicas y paradigmas para minimizar su impacto.

A pesar de que la estimación de proyectos continúa siendo una tarea muy compleja, en muchas ocasiones dejada al albur de la pericia del experto estimador, en las últimas décadas se han desarrollado algunas técnicas para la estimación del esfuerzo de proyectos software completos, tales como casos de usos [3], puntos de función [7] y COCOMO II [4].

Como se ve en un ambiente de que todo debe ser para ayer, se torna caótico el desarrollo de software y de ahí surgen las metodologías ágiles para poder cubrirlo, sin embargo encontrar una combinación de metodología de desarrollo, método de estimación de esfuerzo y costo todavía es un trabajo minucioso y de ahí que surge el área de interés para la elaboración de esta tesis.

1.1. Objetivos.

1.1.1. Objetivo general.

Desarrollo de una propuesta metodológica de estimación de costos incorporando pautas de calidad.

1.1.2. Objetivos específicos.

- Simplificar y disminuir los parámetros o criterios utilizados en COCOMO II para medición del desarrollo de software.
- Ofrecer métodos sencillos y prácticos para obtener calidad en la estimación de un desarrollo de software.
- Desarrollar una metodología de estimación de costos de software para la verificación del nuevo modelo propuesto en este trabajo de investigación en base al COCOMO II.

1.2. Estimación de costo para desarrollo de software.

El producto de software ha tenido una muy alta frecuencia de exceso de calendarización, costos,

problemas de calidad e indiscutibles cancelaciones. Mientras esta mala reputación a menudo es merecida, es importante notar que algunos grandes proyectos de software finalizan a tiempo y con el presupuesto estimado, además funcionan satisfactoriamente cuando éstos son desplegados.

Los grandes proyectos exitosos difieren en muchos sentidos de los fracasos y desastres. Una diferencia importante es, cómo los proyectos exitosos concluyen a tiempo, dentro de los costos, recursos y estimación de calidad prevista en un primer término. De un análisis de resultados de las herramientas de estimación usadas, publicado en [6], el uso de instrumentos de estimación automatizados conduce a estimaciones más exactas. A la inversa, los métodos informales o manuales de llegar en estimaciones iniciales son generalmente inexactos y a menudo excesivamente optimistas.

Las estimaciones de tiempo y costo deberían ser exactas, naturalmente. Pero si ellas difieren de los resultados reales, es más seguro ser ligeramente conservador que ser optimista. Una de las principales quejas sobre los proyectos de software se refiere a su tendencia alarmante de exceder gastos y calendarios planificados.

Debido a que el crecimiento de la estimación de costos es una actividad compleja, existe un crecimiento industrial de compañías dedicadas a ofrecer diferentes marcas comerciales de herramientas de estimación de costos en el mercado. A partir del 2005, algunas de esas herramientas de estimación son: COCOMO II, CoStar, CostModeler, CostXpert, KnowledgePlan®, PRICE S, SEER, SLIM y SoftCost. Algunas de las herramientas de estimación de costos más antiguas, no están activamente en el mercado pero todavía son utilizadas, tales como: CheckPoint, COCOMO, ESTIMACS, REVIC y SPQR/20 [6], ya que su uso no es apoyado por los vendedores, por lo que su utilización está en declive.

1.3. Proceso de estimación.

En la industria en general, es necesario calcular y estimar el esfuerzo y el tamaño del proyecto en etapas muy tempranas del desarrollo del mismo. Sin embargo, si en el ámbito software se hacen las estimaciones en estas fases iniciales, dichas previsiones pueden estar basadas en unos requerimientos erróneos o incompletos, por lo que disminuye mucho su fiabilidad.

El proceso de estimación del coste de un producto software está formado por un conjunto de técnicas y procedimientos que se usan en la organización para poder llegar a una predicción fiable. Este es un proceso continuo, que debe ser usado y consultado a lo largo de todo el ciclo de vida del proyecto.

1.4. Medidas y métricas del software.

En el campo de la ingeniería del software, se suele hablar indistintamente de "métricas" y de "medidas", pero sin embargo existen diferencias entre estos términos. Una medida indica cuantitativamente algún atributo de proceso o de producto (extensión, cantidad, dimensiones, capacidad, tamaño, etc.). Una métrica es definida por el Glosario de Estándares del IEEE (*Institute of Electrical and Electronics Engineers*) como una "medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo determinado" [1].

Si se recopila un sólo tipo de datos, como por ejemplo el número de errores dentro de un componente, se ha establecido una medida. Una métrica de software relaciona de alguna manera las medidas individuales. Siguiendo nuestro ejemplo, podríamos tratar el número de errores encontrados en cada revisión o prueba.

Los ingenieros de *software*, a partir de las medidas, elaboran métricas que les proporcionan información para poder controlar el proceso o el desarrollo de *software*.

2. COCOMO II.

Hoy en día las compañías de *software* desarrollan diferentes programas de *software* en paralelo, que es una tarea muy compleja. Los responsables del proyecto gestionan los diferentes procesos de desarrollo de *software* basados en restricciones como el tiempo, el costo y el número de personal. Calcular el tiempo, el costo y el número de personal es un trabajo muy tedioso para los directores de proyectos de las empresas de *software* en las primeras etapas de la planificación y seguimiento. COCOMO [4] es uno de los mejores modelos para estimar el costo y el tiempo en meses-persona de un proyecto de *software*.

Este capítulo presenta una visión general de los modelos COCOMO que incluye extensiones y modelos independientes, y describe las metodologías subyacentes y la lógica detrás de los modelos y cómo pueden ser utilizados en conjunto para apoyar grandes necesidades de estimación del sistema de software. Concluye con una discusión para unificar estos diversos modelos en un proceso más simplificado que derivará en una nueva metodología completa y fácil de usar.

2.1. Estimación con COCOMO II.

La estructura de COCOMO II se basa en modelos que asumen que se progresa a lo largo de un desarrollo de tipo espiral para consolidar los requisitos y la arquitectura, reduciendo el riesgo; tales modelos son:

– Modelo de Composición de Aplicaciones.

– Modelo de Diseño Temprano.

– Modelo de Arquitectura Tardía.

La estructura de COCOMO II, como se visualiza en la figura 1, se basa en modelos que asumen que se progresa a lo largo de un desarrollo de tipo espiral para consolidar los requisitos y la arquitectura, reduciendo el riesgo; tales modelos son:

– Modelo de Composición de Aplicaciones.

– Modelo de Diseño Temprano.

– Modelo de Arquitectura Tardía.

Figura 1. Arquitectura COCOMO II [4].

COCOMO II utiliza variables establecidas en función de una medida de cinco factores de escala:

PREC Precedencia.

FLEX Flexibilidad de desarrollo.

RESL Resolución de Arquitectura / Riesgos.

TEAM Cohesión de equipo.

PMAT Madurez del proceso.

Para realizar las estimaciones COCOMO II utiliza como medida puntos de objeto, puntos de función o líneas de código, basándose en el diseño lógico del sistema. COCOMO II posee 17 multiplicadores de costos, cada uno de los cuales debe ser estimado:

RELY Fiabilidad.

DATA Tamaño de la Base de Datos.

CPLX Complejidad.

RUSE Reutilización requerida.

DOCU Documentación.

TIME Restricción tiempo de ejecución.

STOR Restricción de almacenamiento principal.

PVOL Volatilidad plataforma.

ACAP Capacidad del analista.

PCAP Capacidad del programador.

AEXP Experiencia de aplicaciones.

PEXP Experiencia plataforma.

LTEX Experiencia del lenguaje y herramienta.

PCON Continuidad del personal.

TOOL Uso de herramientas software.

SITE Desarrollo Multi-lugar.

SCED Planificación requerida.

Una característica importante a destacar es su modelo de reutilización y sus características de auto calibración.

Como se ve, muchos de sus parámetros de configuración son subjetivos, por lo tanto la exactitud de la estimación depende en gran medida de la experiencia de la persona que la realiza, además es muy importante la cantidad de proyectos anteriores (con características similares al nuevo proyecto) porque ayudaría a obtener datos más precisos y partir de una base sólida.

COCOMO II [9] pasó a ser una familia de modelos de productividad, estimación y toma de decisiones. Algunos de los modelos se consideran en desarrollo, es decir que requieren todavía estudios para validarlos y calibrarlos.

3. Metodologías Ágiles.

Hace casi dos décadas que se comenzó a buscar una alternativa a las metodologías formales o tradicionales que estaban sobrecargadas de técnicas y herramientas y que se consideraban excesivamente ?pesadas? y rígidas por su carácter normativo y fuerte dependencia de planificaciones detalladas previas al desarrollo.

Las metodologías ágiles [8] conllevan una filosofía de desarrollo de software liviana, debido a que hacen uso de modelos ágiles. Se considera que un modelo es ágil o liviano cuando se emplea para su construcción una herramienta o técnica sencilla, que apunta a desarrollar un modelo aceptablemente bueno y suficiente en lugar de un modelo perfecto y complejo [2]. Existen actualmente una serie de metodologías que responden a las características de las metodologías ágiles y cada vez están teniendo más adeptos. Aunque los creadores e impulsores de las metodologías ágiles más populares han suscrito El Manifiesto Ágil [10] y coinciden

con sus postulados y principios, cada metodología ágil tiene características propias y hace hincapié en algunos aspectos más específicos.

3.1. Metodologías ágiles de desarrollo más exitosas.

3.1.1. Scrum.

Scrum es un modelo de referencia que define un conjunto de prácticas y roles, y que puede tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará durante un proyecto. En Scrum [5] un proyecto se ejecuta en bloques temporales (iteraciones-*sprints*) de un mes natural (pueden ser de dos o tres semanas, si así se necesita). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto que sea susceptible de ser entregado con el mínimo esfuerzo cuando el cliente lo solicite.

Sprint es el ritmo de los ciclos de *Scrum*. Está delimitado por la reunión de planificación del sprint y la reunión retrospectiva. Una vez que se fija la duración del sprint es inamovible. La mayoría de los equipos eligen dos, tres o cuatro semanas de duración. Diariamente durante el *sprint*, el equipo realiza una reunión de seguimiento muy breve. Al final del *sprint* se entrega el producto al cliente en el que se incluye un incremento de la funcionalidad que tenía al inicio del *sprint*.

El proceso parte de la lista de requisitos priorizada del producto, que actúa como plan del proyecto. En esta lista el cliente ha priorizado los requisitos balanceando el valor que le aportan respecto a su coste y han sido divididos en iteraciones y entregas.

3.1.2. XP (Extreme Programming).

XP [11]es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

3.1.3. Kanban.

Su objetivo es gestionar de manera general cómo se van completando tareas, pero en los últimos años se ha utilizado en la gestión de proyectos de desarrollo *software*.

Las principales reglas de Kanban [8] son las siguientes:

- Visualizar el trabajo y las fases del ciclo de producción o flujo de trabajo.
- Determinar el límite del "trabajo en curso" (WIP - *Work In Progress*).
- Medir el tiempo en completar una tarea (*Lead time*).

3.1.4. Scrumban.

Scrumban [8] es una metodología derivada de los métodos de desarrollo Scrum y Kanban. Es un modelo de desarrollo especialmente adecuado para proyectos de mantenimiento o proyectos en los que las historias de usuarios (requisitos del software) varíen con frecuencia o en los cuales surjan errores de programación inesperados durante todo el ciclo de desarrollo del producto. Para estos casos, los sprints (periodos de duración constante en los cuales se lleva a cabo un trabajo en sí) de la metodología Scrum no son factibles, dado que los errores/impedimentos que surgirán a lo largo de las tareas son difíciles de determinar y por lo tanto, no es posible estimar el tiempo que conlleva cada historia. Por ello, resulta más beneficioso adoptar flujo de trabajo continuo propio del modelo Kanban.

4. Propuesta metodológica.

En este trabajo se presenta un nuevo enfoque para estimar costo de desarrollos de software utilizando metodologías agiles. El objetivo principal de este trabajo es llegar a determinar en etapas tempranas del desarrollo de software la estimación de costos en lo que se refiere exclusivamente al esfuerzo humano en el desarrollo de software incorporando pautas de calidad en la estimación.

4.1. Introducción.

En la actualidad se vive en un mundo de licitaciones y contrataciones de terceros para desarrollos de software y esto aun diciendo que se utiliza un enfoque ágil es imposible no predecir el costo final del producto.

En metodologías ágiles trabajan en equipo entre el principal interesado en el producto, "El Cliente", y las personas que se encargarán de desarrollarlo e implementarlo, "Los Desarrolladores", por ello no suele ser necesario conocer el costo del producto como un todo, debido a que el cliente conoce los detalles de la metodología.

Sin embargo, la necesidad de ambas partes de conocer lo que costará el producto es de suma importancia, de ahí el énfasis de esta tesis en solucionar esta brecha y resolver el problema.

Se propone así una estimación orientada a metodologías agiles considerando calidad en la esti-

mación con los factores de escala y los multiplicadores de esfuerzo de COCOMO II y eliminando el conteo de líneas de código (LOC).

4.2. Planificación.

A continuación se describe el marco de trabajo que compone una serie de etapas, que definen pasos preestablecidos y procedimientos que se siguieron para componer la nueva metodología de estimación de desarrollo de *software*.

Las definiciones se basaron en seis etapas que aseguran el cumplimiento del objetivo:

- Análisis de parámetros.
- SCRUM, como gestión del proyecto.
- COCOMO II, como pauta de calidad.
- Definición de la propuesta.
- Formulas y cálculos resultantes.
- Evaluación de la propuesta.

4.2.1. Etapa 1 - Análisis de Parámetros.

Antes de comenzar a comentar los parámetros seleccionados para la propuesta, se procede a una breve comparación de los parámetros utilizados en metodologías tradicionales para estimar con COCOMO II y aquellos parámetros que se puede disponer utilizando SCRUM como gestión del proyecto de desarrollo de *software*.

COCOMO II usa Puntos de Función como método de estimación para desarrollos de software orientados a objetos, en la tabla 1 se observa la comparación entre parámetros de COCOMO II y parámetros manejados en SCRUM.

Tabla 1. Parámetros de COCOMO & SCRUM (fuente propia).

COCOMO II	SCRUM
Casos de Usos	Historias de Usuarios
Número de Entradas de Usuarios	
Número de Salidas de Usuarios	
Número de Peticiones de Usuarios	
Número de Archivo de Usuarios	MER - Diagrama de Modelo Entidad - Relación
Número de Interfaces de Usuarios	Prototipos de Interfaz
Factor de Ponderación (Simple—Medio—Complejo)	Complejidad de cada Historia
Puntos de Función a Líneas de Códigos	Suma de Complejidades de Historias
Factores de Costos	Porcentaje de dedicación
Multiplicadores de Esfuerzos	
Esfuerzo medio en Meses Personas	Sprint Iteraciones

Actualmente no se cuenta con ajustes ni calibraciones para convertir puntos de historias estimadas con base a complejidades a LOC, aunque tampoco ese es el objetivo ya que líneas de código en desarrollo de software orientado a objetos se quedaron obsoletas debido a que esto depende del estilo y conocimiento del programador.

4.2.2. Etapa 2 ? SCRUM, como Gestión del Proyecto.

Se ha elegido SCRUM como gestión de proyecto ágil por ser un modelo de referencia a nivel internacional. SCRUM se define como un conjunto de prácticas, roles y parámetros útiles como punto de partida para la definición de la nueva metodología de estimación de costo de desarrollo de software.

A continuación se cita los artefactos y herramientas utilizados para la nueva propuesta.

1. Pilas de Productos.
2. Historias de Usuario.
3. Complejidad de cada Historia.
4. Prioridad por cada Historias.
5. Sprint.

4.2.3. Etapa 3 - COCOMO II, como Pauta de Calidad.

El COCOMO II usa cinco factores de escala (SF) que servirán para ajustar las economías y deseconomías de escala que se presentan en proyectos de software de diferentes tamaños. A continuación se muestra la fórmula de la ecuación del cálculo los factores de escala.

$$E = B + 0{,}001. \sum_{j=1}^{5} SF_j \qquad (1)$$

En esta fórmula B es una constante, el valor para este parámetro fue obtenido en la calibración de 161 proyectos en las base de datos de COCOMO II y es inicialmente igual a 0.91 [4].

Cuando el valor 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 E es mayor que 1 el proyecto muestra una deseconomía de escala lo que doblaría el tamaño y podría provocar un aumento de más del doble en el esfuerzo. La sumatoria $\sum$ consiste en 5 factores de escala: Precedencia, Flexibilidad de Desarrollo, Resolución de Riesgos y Arquitectura, Cohesión del Equipo y Maduración del Proceso.

Además, COCOMO II cuenta también con 17 multiplicadores para el diseño Post-Arquitectura.

Los multiplicadores de esfuerzo en COCOMO, son características del proyecto que tienen un efecto lineal en el esfuerzo de un proyecto. Los multiplicadores de esfuerzo se dividen en varias partes como:

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).

4.2.4. Etapa 4 - Definición de la Propuesta.

Se presenta esta nueva metodología de estimación de costos definida bajo el enfoque de metodologías ágiles, que sirve para estimar proyectos a través de métodos sencillos y aplicables a equipos de desarrollo ágil. La misma ofrece estándares, recomendaciones y artefactos que requiere una metodología de estimación de costos.

Se define esta arquitectura (figura 2), con su principal objetivo, "la estimación de costo" al proceso de desarrollo de software y, en segundo lugar la planificación de los tiempos de las diferentes actividades necesarias en el proyecto.

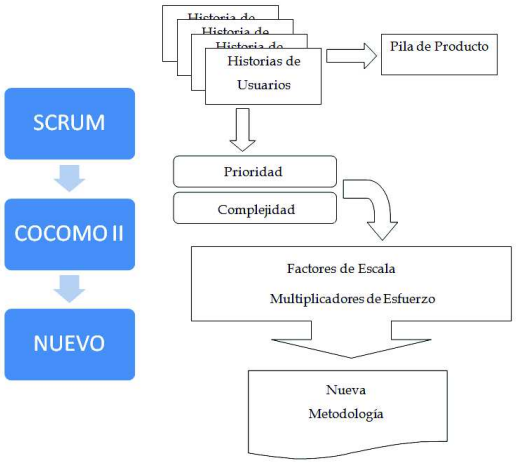

Figura 2. Nueva Arquitectura (fuente propia).

4.2.5. Etapa 5 - Fórmulas y Cálculos Resultantes.

Para cuantificar el desarrollo de software, se necesita fijar medidas de magnitudes para obtener los resultados precisos, a partir de las cuales se evalúen los resultados, la nueva metodología propone fórmulas que se detallan a continuación para lograr la estimación de costos.

Esta metodología de medición puede dividirse en diez pasos de cálculos:

1. Determinar el conteo de puntos de historias.
2. Determinar la duración del sprint.
3. Determinar la velocidad del equipo.
4. Determinar el número de personas en el equipo.
5. Determinar el factor de escala del proyecto.
6. Determinar los multiplicadores de esfuerzo del proyecto.
7. Determinar los puntos de historias ideales para el sprint.
8. Determinar los puntos de historias ajustados para el sprint.
9. Determinar la cantidad de iteraciones para el proyecto.
10. Determinar el costo final del desarrollo de software.

A continuación se detallan cada una:

1. Total de Puntos de Historias (**TStoryPoint**).
2. Duración de Sprint (**DSprint**).
3. Velocidad Inicial del Equipo (**V**).
4. Nº de Personas en el Equipo (**N**).
5. Puntos de Historias del Proyecto (**StoryPoint**).

   $$\textbf{StoryPoint} = V * \textbf{DSprint}$$

6. Exponente de Factor de Escala (**E**).

   $E = B + 0{,}001. \sum_{j=1}^{5} SF_j$

7. Multiplicadores de Esfuerzo (**EM**).

   $\prod_{i=1}^{n} EM_i$

8. Puntos de Historias Ajustadas (**StoryPointAjust**)

   $$\textbf{StoryPointAjust} = \textbf{StoryPointAjust} \prod_{i=1}^{n} EM_i$$

9. Cantidad de Iteraciones para el proyecto (**I**)

   $$I = \{\frac{TStoryPoint}{StoryPointAjust}$$

10. Costo promedio diario del equipo (**CostDay**).
11. Estimación de costo del desarrollo de software (**CostDev**)

    $Cost = ((CostDay * DSprint) * I * N)$

4.2.6. Etapa 6 - Evaluación de la Propuesta.

Para evaluar la propuesta se procede a definir ciertas características que deben contemplar una metodología de estimación de costos clasificada de manera esencial a medir el desarrollo de software.

La calidad de las medidas debería facilitar el desarrollo de modelos que sean capaces de predecir el comportamiento de determinados parámetros que afectan al desarrollo de software. Una metodología de medida ideal debería ser:

- Objetiva.
- Sencilla, definible con precisión para que pueda ser evaluada.
- Fácilmente obtenible (a un coste razonable).
- Válida, la métrica debería medir exactamente lo que se quiere medir y no otra cosa.
- Robusta. Debería de ser relativamente insensible a cambios poco significativos en el proceso o en el producto.

5. Pruebas experimentales.

5.1. Casa Práctico 1.

El primer caso práctico es evaluar el software existente con la metodología normal de COCOMO II, comparar los resultados obtenidos a base de puntos de función y a base de Líneas de Código, esto sirve como aporte a la empresa para conocer el valor del software SGPTI (Sistema de Gestión Parque Tecnológico de Itaipú), como se le denomina al producto, según el modelo COCOMO II.

Para conocer acerca de la realidad de cómo se gestionó el desarrollo del software SGPTI, antes de comenzar con la estimación se elaboró un cuestionario con el fin de tener en claro las técnicas que fueron utilizadas por el parque y así completar los factores de escala y multiplicadores de esfuerzo antes de la estimación de software.

5.2. Caso Práctico 2.

El segundo caso práctico ha sido probar la nueva metodología propuesta en esta tesis, estimando la reingeniería de un módulo del software del parque utilizando la nueva metodología y así comprobar si contempla los requisitos citados en la Etapa 6 de la propuesta.

6. Resultados obtenidos.

6.1. Caso Práctico 1.

El tamaño del software SGPTI realizado por el modelo USC COCOMO II.2000.4 ha sido de 252.598 LOC y la productividad resultante en meses persona ha sido 463.5 PM que básicamente constituye un promedio de 3 años y medio de proyecto con una cantidad promedio de diez personas trabajando en el equipo. Se puede deducir que el proyecto del primer caso práctico es PRODUCTIVO bajo los cálculos estimados de COCOMO II; a continuación se listan los posibles motivos por los cuales se generaron dichos resultados:

- El nivel de organización del Jefe de Proyecto, este 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.
- 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 se presume que los tiempos no son muy precisos.

6.2. Caso Práctico 2.

En el segundo caso práctico se observa que el proyecto tendrá un costo de 47.072.032 Gs. para un equipo de seis personas para culminar con dos iteraciones de Sprint que duran 20 días cada una. Se debe considerar que hubo un análisis previo de lo que desea el cliente y luego de la formulación de la plantilla 3, se procedió al cálculo de la estimación.

Se pueden manipular los resultados de acuerdo a los factores de escala y multiplicadores de esfuerzo como se observa en la tabla 2.

Tabla 2. Tabla de valores de factores y multiplicadores (fuente propia).

		Costo	Iteraciones
TEAM	**ALTA**	37.454.374	1.56
	NOM	40.428.713	1.68
	BAJA	47.072.032	1.82
MATURITY	**ALTA**	41.741.521	1.73
	NOM	37.454.374	1.56
	BAJA	33.607.548	1.40
PLEX	**ALTA**	31.269.248	1.42
	NOM	37.454.374	1.56
	BAJA	40.098.213	1.67

Como se observa en la 2 el factor de escala **TEAM**, la que se refiere a la cohesión del Equipo, modificando este valor para NOMINAL el costo varía como se visualiza en la figura 3.

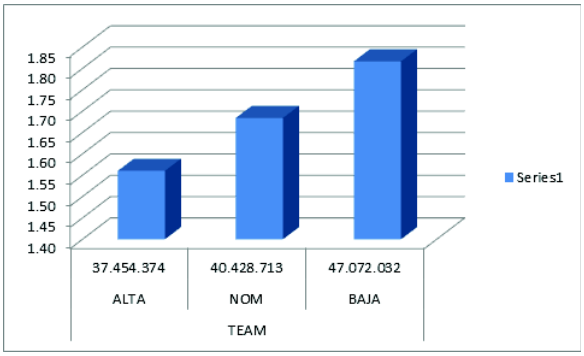

Figura 3. Factor TEAM (fuente propia).

Otro factor importante es el **MATURITY**, podemos ver que si cambiamos este valor a ALTA el costo sube a 41.741.521 Gs (figura 4).

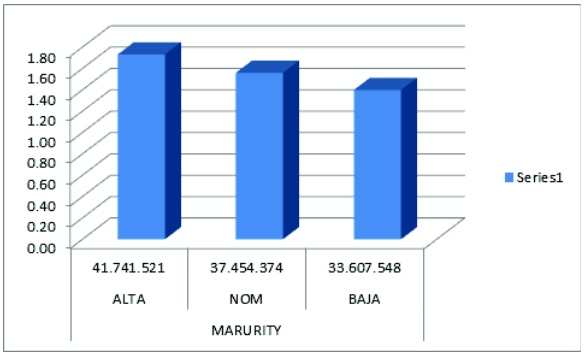

Figura 4. Factor MATURITY (fuente propia).

Entre los multiplicadores de esfuerzo se analiza el PLEX, que se refiere a experiencia en la plataforma, si modificamos este valor a BAJA podemos observar un aumento en el costo a 40.098.213 Gs (figura 5).

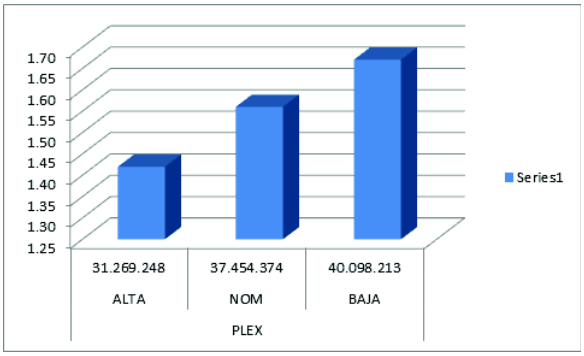

Figura 5. Multiplicador de Esfuerzo PLEX (fuente propia).

Como se observa en los análisis todos estos factores y multiplicadores cumplen un papel fundamental a la hora de estimar el costo.

7. Conclusión.

7.1. Principales logros.

Por los resultados obtenidos de la estimación con la nueva propuesta metodológica a proyectos reales de software, se puede confirmar la hipótesis planteada, esto es que, la utilización de metodologías ágiles en el desarrollo de software puede llegar a ser cada vez más preciso en cuanto a estimación de tiempo y costo. Aunque no puede generalizarse, esto está cada vez más estable de acuerdo a las prácticas actuales en el desarrollo de software.

La investigación de campo realizada en la Fundación Parque Tecnológico de Itaipú ayudó a definir la situación real y actual sobre cómo manejan la estimación en las empresas, se comprobó que no usan técnicas, ni metodología aplicable para el tipo de software a desarrollar, por lo general se basan en la experiencia del profesional del equipo de trabajo. Por esta razón la falencia en la estimación temprana produce desfases en la entrega del producto.

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

Esta metodología propuesta apoya en la medición temprana y logra definir el esfuerzo, tiempo y costo necesarios para cumplir con las entregas del producto al finalizar cada ciclo del Sprint.

La metodología puede objetarse por estar relacionada a una metodología ágil particular de desarrollo y en consecuencia no podrán estandarizarse, sin embargo, casi todas utilizan casi los mismos parámetros y se adecuan a la realidad actual para obtener una estimación real.

Se requiere prototipar las historias de usuarios para conseguir una mejor definición y una mejor ejecución del plan.

Para realizar la estimación de complejidad de historias debe existir un conceso dentro de la organización para llegar a lo más exacto posible para cada historia de usuario, y así lograr que la medición se acerque lo máximo a la realidad. La correcta redivisión de los factores de escala y los multiplicadores de esfuerzo varían notablemente en el resultado final, se requiere buena disposición y trabajo en equipo para consensuar cada valor correspondiente, y por consiguiente proveer calidad a la estimación de costos.

Con esta metodología se puede estimar desarrollos de software en menor tiempo, definiendo la arquitectura y trabajando en paralelo con los requisitos que se van sumando al proyecto.

Gracias a esta metodología el equipo podrá demostrar el trabajo diario y entregar el producto con calidad y una precisión de tiempo exitosa, a

su vez la empresa en cuestión valorará el pago correcto al equipo por los resultados más reales en la estimación.

Se demostró la problemática que se vive en la actualidad en equipos de desarrollos de software por no contar con una metodología de estimación de costo.

7.2. Trabajos futuros.

A los efectos de validar, de forma empírica, la nueva metodología que se presenta en este trabajo, es necesario obtener más confirmaciones experimentales de la efectividad de la propuesta realizada, tanto cualitativamente como cuantitativamente. Es decir, cotejar mediante un mayor trabajo de campo para corroborar así la eficacia y la eficiencia de la nueva metodología propuesta y enriquecerla con los ajustes necesarios si fuera conveniente.

Otra línea de investigación interesante, sería el análisis y la especificación formal de un modelado y prototipado para la posterior implementación de la nueva propuesta metodológica.

Referencias bibliográficas

[1] Agarwal, R.Y. Estimating software projects. Software engineers Notes, 2001.

[2] Ahmed E. Hassan, R. C. The Chaos of Software Development. In IEEE Computer Society, 2003.

[3] Banerjee, G. Use Case Point - An Estimation Approach, 2001.

[4] Boehm B., C. A. Software Cost Estimation with COCOMO II. Prentice-Hall,2000

[5] Cockburn, A. Agile Software Development The cooperative Game Second Edition. Boston. USA.: Addison Wesley,2007.

[6] Jones, C. Applied Software Measurement . McGraw Hill, 2008.

[7] Longstreet, C. I). Function Points Analysis Training Course. Retrieved from `http://www.softwaremetrics.com/Function%20Point%20Training%20Booklet%20New.pdf`, 2004.

[8] Mendes Calo, K. E. Evaluación de Metodologías Ágiles para Desarrollo de Software. XII Workshop de Investigadores en Ciencias de la Computación,2010.

[9] Milicic, D. Applying COCOMO II, A case study. Master Thesis Software Engineering,2004.

[10] The Agile Manifesto. The Agile Manifesto. Retrieved from `http://agilemanifesto.org/authors.html`, 2001

[11] Wells, D. Extreme Programming. Retrieved from `www.extremeprogramming.org,www.xprogramming.com`, 2013.