Reducción del Proceso de Desarrollo de Software basado en CMM

EN EL NIVEL 3 (REPETIBLE)

Francisco Álvarez [1] - Alfredo Weitzenfeld [2]

RESUMEN

El objetivo de este trabajo es mostrar la aplicación de un modelo de desarrollo de software orientado en ciclos cortos basado en CMM (Capability Maturity Model), para productos de software pequeños y equipos de desarrollo con pocos integrantes. El modelo considera los principales aspectos en la planeación, control y dirección del proceso orientado a la calidad de software logrando el nivel 3 del CMM (se comprueba a través de las KPA's cubiertas). Se aplicó en un producto de software (Sistema de Control Médico) obteniendo resultados favorables como un control adecuado del proceso de desarrollo y tiempos estimados de desarrollo muy cercanos a los de ejecución.

PALABRAS CLAVE:

Calidad de Software, Procesos de desarrollo, CMM (Capability Maturity Model), KPA's (Key Process Areas).

INTRODUCCIÓN

La industria del software se considera reciente de tan sólo unas cuantas décadas, esta ha presentado un gran desarrollo y avance. La industria se ha conducido como especial, debido en gran medida a las ganancias económicas que representa (muchos gobiernos hacen esfuerzos importantes por atraer inversiones de este tipo a sus países) y el mismo tipo de producto que desarrolla (por muchos años el desarrollo de software se ha considerado como algo artesanal).

No es de extrañarse que se desee producir más y con mejores niveles de calidad, tal como se plantea con otros tipos de productos, sin embargo la solución a esto no ha sido tarea fácil, se han presentado distintas opciones en la mejora y corrección tanto de los procesos de desarrollo como de los mismos productos generados.

De lo anterior que se considere como una de las soluciones más viables el control de los procesos de desarrollo, para lograr una mejora en los mismos y en los productos generados por éstos, que permita una implementación sencilla tomando en consideración los aspectos relevantes y exitosos de las prácticas de producción de las empresas generadoras de software.

En general el éxito de los proyectos de software dependen de la buena planeación y control de su proceso de desarrollo; de aquí que se perciba la necesidad de continuar con modelos de calidad que permitan planificar, controlar y mejorar procesos [3].

De las soluciones más completas y probadas en la búsqueda de generación de productos con calidad de software, es a través del control de los procesos de producción, la calidad de un sistema de software es gobernada por la calidad del

1 Departamento de Sistemas Electrónicos, Universidad Autónoma de Aguascalientes (UAA), fjalvar@correo.uaa.mx

2 Departamento de Ciencias Computacionales, Instituto Tecnológico Autónomo de México (ITAM), alfredo@itam.mx

proceso utilizado para desarrollarlo y mantenerlo [10]. Los modelos de procesos definen un orden para llevar a cabo los distintos aspectos de este, llevan pautas como: planeación, predicción, evaluación, etc. [6].

Un proceso bien conocido, de difusión amplia y utilización práctica, sustentada en medición y predicción de eventos, permite controlar en buena medida la producción de software [5]. La evaluación de los procesos evita la producción de software con especificaciones incompletas o anómalas, la aplicación incorrecta de metodologías, etc. [5]. Esto soluciona uno de los problemas planteados inicialmente: *la predicción del tiempo de desarrollo de proyectos, así como la congruencia del producto generado con ciertas expectativas por parte del cliente*.

Uno de los modelos que permite lograr el control y la predicción sobre los procesos, es el CMM [14] a través de un marco de trabajo representado por guías de recomendaciones para organizaciones generadoras de software que quieren incrementar su capacidad de procesos de software, considerando los siguientes aspectos:

1. Identificación de fortalezas y debilidades en la organización.
2. Identificación de los riesgos de seleccionar entre diferentes contratos y seguimiento de los mismos.
3. Entendimiento de las actividades necesarias para planear e implementar los procesos de software.
4. Ayuda en la definición e implementación de procesos de software en la organización a través de guías.

El CMM tiene como objetivo evaluar los procesos en sus distintos niveles de madurez, a través de la identificación de los niveles por los cuales una organización debe escalar para establecer una cultura de excelencia en la Ingeniería de Software, partiendo de la premisa: *procesos sin una apropiada fundamentación fallan*.

Los procesos son evaluados a través de distintos niveles de madurez, que van desde prácticas

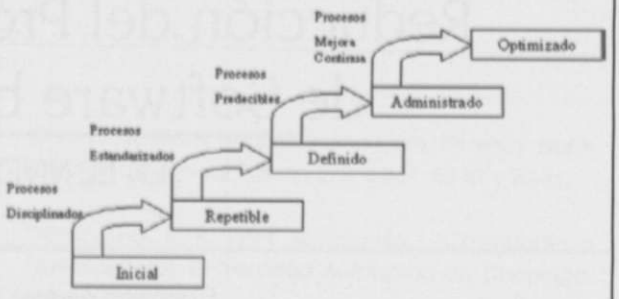

Figura 1. Los cinco niveles de madurez del proceso de software.

desordenadas o descoordinadas, hasta lograr una mejora continua de procesos y por ende de productos.

La madurez de los procesos es determinada por las distintas KPA's, es decir, de las áreas básicas que componen a las organizaciones y su evolución: a) Definir los procesos para tener la capacidad de repetirlos, b) Estandarizar los procesos para definirlos en toda la organización, c) Predecir los procesos para administrarlos (planificarlos, organizarlos, dirigirlos, controlarlos) y por último d) Mejorar continuamente los procesos para optimizarlos (hacerlos eficientes y eficaces).

El problema general con CMM es que sus niveles son a veces inadecuados para organizaciones cuyo desafío principal es salir rápidamente al mercado con productos innovadores o con reacciones inmediatas del entorno, esto principalmente al enfoque de estructura de proyectos en desarrollo grandes, de aquí que la solución a implementar el modelo CMM en proyectos cortos y organizaciones de igual tamaño sea de suma utilidad para este tipo de empresas.

El PEMM (Performance Engineering Maturity Model) [21] presenta un modelo para evaluar el nivel de integración y aplicación de ejecución de diseño, llamado ingeniería de la ejecución del modelo de madurez. Se apoya en el modelo de madurez de capacidades de CMM.

El objetivo del PEMM es evaluar la ejecución de la Ingeniería procesada así como la integración del proceso. El modelo de evaluación propuesto puede ser aplicado en evaluar una organización, así como en propios desarrollos de procesos específicos.

El PEMM puede utilizarse en la selección de criterios para escoger un proveedor de sistemas de software para los productos críticos o semi-críticos en una organización.

El método base de la medición del proceso de PEMM es la Meta-Pregunta-Métrica (GQM, Goal Question Metric) que se usa para identificar y medir los procesos de desarrollo. El modelo al igual que el CMM, cuenta con 5 niveles, los cuales determinan: la mejora del comportamiento de ejecución y el decremento del riesgo de ejecución a través de estos niveles.

La evaluación de una compañía se hace a través de la medición de sus aspectos generales, la organización, la definición de procesos de ingeniería, el proyecto de la dirección y la tecnología, a través de 34 preguntas (GQM).

Lo acertivo de este modelo es la evaluación de la maduración de procesos en la misma organización tomando aspectos tecnológicos relevantes como el software para la definición y el diseño de procesos de desarrollo.

SPICE [5], es un modelo internacional sobre el cuál se están basando algunos estándares de certificación, incluso CMM tiende a éste a través del modelo desarrollado CMMi (Capability Maturity Model Integrated).

Este modelo fomenta productos de calidad, promueve la optimización de procesos, mejoras en procesos, cultura tanto a nivel directivo como en el nivel de usuarios, facilita la evaluación del producto a través de los procesos de desarrollo.

Tiene tres características principales: *El marco de valor* que contempla una dimensión funcional del proceso, la evidencia para la evaluación y la recurrencia dada por la selección de instancias de proyectos o producto.

SPICE permite entre otras cosas, que la gerencia se asegure que el proceso se encuentra alineado con las necesidades del negocio, que los proveedores de software tengan que someterse a una sola evaluación para aspirar a nuevos negocios, que las organizaciones de software dispongan de una herramienta universalmente reconocida para dar soporte a su programa de mejora continua.

El modelo establece un común denominador para una evaluación uniforme de los procesos de software. Es de vital importancia atender de manera especial cada uno de los requerimientos del modelo SPICE haciendo hincapié en la calidad y actualización, así como en la vigencia del producto. Ya que la tecnología es cambiante, las fases que marca el modelo SPICE son sin duda uno de los pilares en que se tendrá que trabajar con la mayor dedicación para obtener calidad en el producto y el servicio del mismo sea excelente, además de generar la confianza necesaria hacia la dirección y hacia el usuario de donde se obtiene la información.

Existen algunos problemas a resolver, relacionados con los modelos de madurez de procesos en relación a su propia naturaleza e implementación:

1. Los modelos de alta difusión como CMM, tienen el principal inconveniente de ser modelos para proyectos de gran tamaño, de aquí lo complicado de su generalización en proyectos de menor formato.
2. Muchos modelos no convergen sus objetivos con los de la organización, por esta razón es complicada la aceptación e instauración de los mismos.
3. Algunos modelos caen en una definición excesiva de procesos, es decir se pueden presentar reglas, actividades, normas, etc. que pretenden guiar a detalle, complicando su aplicación o entendimiento.
4. Casi ningún modelo hace énfasis en la certificación o evaluación de proyectos de pequeño y/o mediano formato.
5. Existe un decremento de la productividad al inicio de la aplicación del modelo.
6. Se pueden interpretar los modelos como guías detalladas.
7. Resistencia de la organización al cambio (culturas organizacionales arraigadas).

CONSIDERACIONES EN LA REDUCCIÓN DE CICLO DE DESARROLLO EN PROYECTOS Y EQUIPOS PEQUEÑOS

Para adoptar el CMM en un mayor número de organizaciones y tamaños del proyecto, CMM presenta una serie de aspectos por resolver [4]:

1. La excesiva documentación de todo el proceso y sus aspectos relacionados.
2. La administración de KPA's orientada a grandes organizaciones.

3. Los recursos limitados.
4. Los costos de entrenamiento.
5. Las prácticas sin relación al tipo de proyecto.
6. La falta de guía de las necesidades del proyecto y equipo de desarrollo.

Estos aspectos están estrechamente ligados a la propia filosofía del modelo: organizaciones complejas con un gran número de integrantes, así como proyectos complejos por desarrollar.

En equipos pequeños la calidad del propio equipo es de suma importancia para resultados favorables, ya que altos niveles de habilidades y experiencias generan calidad en los productos [16], por otro lado generalmente este equipo de desarrollo no puede dedicar mucho tiempo a procedimientos administrativos o de documentación. El tiempo es dedicado en gran parte al diseño, programación y pruebas del producto (construcción del software).

Otras investigaciones mencionan que ajustar procesos de desarrollo a través de reducir el ciclo en organizaciones pequeñas mejora los resultados de los proyectos [9]. El modelo de PROCESSUS [9] involucra actividades como análisis, definición, entrenamiento, divulgación y preparación desde la fase de la introducción del proceso. Esto es seguido por la fase de definición del proceso y fase de optimización del proceso. Su estudio indica que la documentación del proceso y documentación del proyecto mejoran la calidad de los productos.

El PSP (Personal Software Process) [10] y TSP (Team Software Process) [11] son de alguna manera la respuesta en la solución a las dos grandes limitantes mencionadas: tiempo e implementación, sin embargo no se tiene una evidencia completa en los resultados generados por ambas técnicas. Los resultados observados demuestran que los proyectos generados bajo cualquiera de las técnicas, permite una disminución en tiempo y un control en el proceso de desarrollo [13][7].

En relación a algunas evidencias empíricas sobre la aplicación de CMM en equipos de desarrollo muy pequeños y proyectos cortos, se contempla un estudio de una aplicación del CMM a un equipo con estas características (menos de 10 personas) y con recursos muy limitados, el estudio presenta que el equipo pudo llegar al segundo nivel de CMM: Nivel Repetible [1]; el logro alcanzado se midió a través de las KPA's cubiertas de este nivel. Por otro lado se reporta que la medición de la mejora del proceso de desarrollo, implica una dificultad, ya que es complicado conocer qué medir [2].

Un objetivo que se presenta en el anterior estudio es que CMM podría adaptarse a equipos muy pequeños para mejoras de procesos de software (SPI). La forma en la que este adopta CMM es a través de lograr los objetivos de las áreas del proceso más importantes KPA's (Key Process Areas) sin embargo se presenta el inconveniente de que CMM no puede aplicarse de una manera directa para equipos pequeños y se hace necesaria una dirección del equipo para ajustar el proceso.

Otro estudio con equipos de tamaño semejante ajusta el Modelo de Madurez de Capacidades (CMM) en proyectos pequeños a través de un marco de trabajo del proceso [12]. Los resultados se reportan en la reducción de documentación y el esfuerzo invertido, manteniendo la calidad del software, es decir se reduce el ciclo a través de la disminución de los formatos de control y documentación en proyectos cortos.

Como puede observarse estos casos presentan evidencias sobre la aplicación asertiva de CMM en proyectos y equipos pequeños, aunque todavía no se presenta una solución completa a un ajuste necesario para el modelo, los factores que consideran la reducción del proceso son:

1. Disminución de las fases considerando las necesidades de un proyecto corto.
2. La consideración de prioridad de actividades de equipos pequeños de desarrollo enfatizando en la construcción del software.
3. Control del proyecto y disminución de tiempos de implementación de CMM a través de PSP y TSP.
4. Lograr los objetivos de las KPA's indispensables en la definición de procesos de proyectos cortos.
5. Disminución de los formatos de control y documentación para facilitar la administración del proyecto.

Con estas experiencias de casos de estudios reportados se hace evidente la creación de un modelo de procesos de desarrollo en este tipo de proyectos y equipos a través de la reducción del ciclo de desarrollo.

MODELO REDUCIDO PROPUESTO

El modelo propuesto involucró el desarrollo de 12 actividades:

1. Compromiso de la dirección.
2. Establecimiento del SPEG (Software Process Engineering Group).
3. Metas propuestas.
4. Creación de una guía.
5. Plan de información sobre CMM.
6. Desarrollo de un proceso de software común.
7. Proceso trazado por CMM.
8. Entrenamiento.
9. Recolección de los datos y mejora del proceso.
10. Desarrollo del proceso pequeño (ajustado).
11. Valoración del SQA (Software Quality Assurance).
12. Mejora continua.

Ciclo de desarrollo reducido para proyectos pequeños

El ciclo de vida del proyecto mantiene una base categorizando y controlando las varias actividades exigidas a desarrollar en un producto de software. El ciclo de vida del proyecto para un proyecto pequeño es similar al modelo de cascada pero el ciclo entero se estructura en tres fases: planeación, construcción e instrumentación, con lo cual se obtiene la primera reducción del ciclo de desarrollo (ver Tabla 1). Como el número de fases es reducido, también se reduce el esfuerzo invertido en el seguimiento del proyecto, la revisión de SQA y la recolección de los datos.

TABLA 1.-
Fases en procesos pequeños y procesos estándar.

Fases procesos pequeños	Fases en procesos estándar
1. Plan	1. Administración de
2. Construcción.	requerimientos.
3. Implementación.	2. Planeación de recursos.
	3. Diseño de especificaciones.
	4. Planeación del proyecto de software.
	5. Desarrollo.
	6. Aceptación.
	7. Realización.
	8. Mantenimiento.

La Fig. 2 muestra las fases, subfases y aportación del SQA en todo el proceso de desarrollo. El procedimiento de dirección de configuración se aplica en todas las fases.

El costo de entrenamiento es un problema crítico del proyecto, sobre todo para los proyectos pequeños [2]. Debido a los recursos limitados, los entrenamientos formales no siempre pueden ser posibles. A veces, también es imposible esperar el entrenamiento antes de iniciar el desarrollo real de un proyecto.

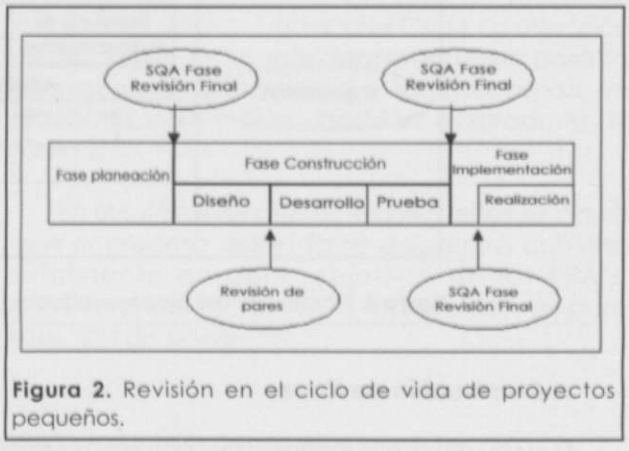

Figura 2. Revisión en el ciclo de vida de proyectos pequeños.

Cuando los recursos humanos están normalmente limitados en organizaciones pequeñas, el personal puede ser involucrado simultáneamente en varios proyectos con una base de tiempo por partes, y puede tomar el mismo proyecto a través de roles múltiples. Deben definirse reglas para evitar conflictos debido a estos roles múltiples [10]. Hay restricciones en los roles de SQA y miembros de equipos de prueba. Puesto que la función del SQA debe ser independiente del desarrollo del software que agrupa, el personal de SQA no debe involucrarse en desarrollos propios.

REDUCCIÓN DEL CICLO EN EL CASO DE ESTUDIO

Considerando los aspectos revisados en los apartados anteriores se presenta la implementación de los mismos en un proyecto denominado: *Sistema de Control Médico*.

Algunos datos del proyecto:

- Período desarrollo: Agosto 2002 - Diciembre 2002.
- **Lugar:** Estado de Aguascalientes.

- **Características:** Aplicación para la administración de registros médicos, registros de pacientes, historiales de registros médicos, etc. con un equipo de desarrollo pequeño (4 ingenieros).
- **Observaciones:** Al iniciar los proyectos no se contaba con una experiencia sobre modelos de madurez de procesos, por lo que se capacitó al equipo de desarrollo y se dio seguimiento a la correcta aplicación del modelo reducido. La capacitación aproximadamente fue de 60 horas.

Considerando el ciclo de vida presentado en la sección anterior, se muestra a continuación el framework (ver Fig. 3) del proceso reducido incluyendo las técnicas utilizadas, así como los resultados obtenidos del mismo.

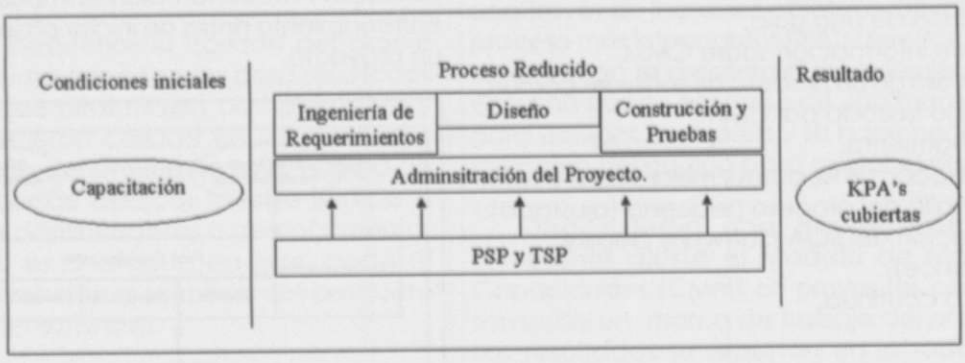

Figura 3. Framework del proceso reducido de software.

a. Disminución de fases.

El ciclo de vida del proyecto corto es similar al modelo de cascada, ya que las fases se realizan una a una, debido a las dimensiones del proyecto. El ciclo entero se estructura en tres fases [15]: planeación, construcción e implementación con lo cual se obtiene la primera reducción del ciclo de desarrollo (ver tabla 1). Debido al número de fases reducido, el esfuerzo de administración del proyecto disminuye.

Esta reducción se debe principalmente a que la parte central del proceso es la construcción del software [16], sin embargo se añaden las fases de planeación para lograr un mejor control del proyecto y de implementación para incrementar la aceptación del producto.

En el proyecto se llevaron las fases reducidas considerando la construcción del software (requerimientos, diseño, codificación y pruebas) y la administración del proyecto (planificación, dirección y control), logrando cubrir así la parte central del proceso de desarrollo.

La ejecución de las fases en el proyecto, "Sistema de Control Médico" se describen a continuación:

1. Requerimientos. Se utilizaron varias técnicas para la obtención y administración de los requerimientos del sistema tratando en todo momento de tener contacto estrecho con el cliente y los usuarios a través de las técnicas diseñadas: "Determinación de Objetivos", "Delimitación del Sistema a Construir", "Técnica de acercamiento a la entrevista", "Diagramas de Casos de Uso (UML)". Los productos de estas técnicas fueron validadas en todo momento tanto por el director del proyecto como por los clientes y usuarios.

2. Diseño. Dando el seguimiento se obtuvieron los "Diagramas de Clases" (UML) y de "Diagramas de Secuencias" correspondientes, así como el "Diseño de Interfaces de Usuario".

3. Construcción y pruebas. La construcción y algunas de las pruebas del sistema se desarrollaron en paralelo y al final las pruebas ejecutadas fueron de: "Integración del Sistema", "Almacenamiento", "Factores humanos".

4. Administración del proyecto. La fase de administración involucró planeaciones y controles grupales, así como planeaciones y controles individuales, considerando en la

planeación grupal: "Análisis de Riesgos", "Análisis de Escenarios" para la solución a posibles problemas a enfrentar, "Controles de Recursos, Tiempos y Actividades ejecutadas vs. Recursos, Tiempos y Actividades planificadas". Por último se evaluaron los objetivos alcanzados en el producto con el cliente.

b. Control del proyecto e implementación de PSP y TSP.

Uno de los grandes aciertos que tiene el PSP y TSP es la facilidad de implementación en las organizaciones [8], sin embargo se debe tener un plan de capacitación y ejecución de las técnicas (muchas de las veces las formas de registro pueden parecer excesivas en un inicio). La ventaja principal es la disciplina de los ingenieros en el registro de sus actividades para lograr una mejor planificación y ejecución de sus proyectos, así como el incrementar la calidad de sus productos.

En el proyecto ejecutado, se inició con una capacitación en la utilización de PSP y TSP, sin embargo al momento de la utilización de algunos registros los integrantes se mostraron algo indispuestos a su aplicación de aquí que se tratara de sólo utilizar las formas representativas de las dos técnicas. Por ejemplo en la asignación de roles y estimaciones de esfuerzos individuales lograron ser muy próximos a la ejecución.

c. Cobertura de KPA's en el proceso de desarrollo de software reducido.

Se ha comparado el proceso de desarrollo reducido contra las 14 KPA's del CMM. La selección de las KPA's es la más representativa para la consecución del nivel 2 y la aplicación en proyectos cortos. El resultado es mostrado en la Tabla 2.

En el estatus de proyectos pequeños se indica que el proceso ajustado se encuentra cubierto totalmente en ocho objetivos de las KPA's, considerando ser las KPA's representativas para este tipo de proyecto.

TABLA 2 .- Comparación con CMM en relación a las KPA'S en el estatus de pequeños proyectos.

Key Process Areas (áreas claves de procesos).	% de satisfacción
Administración de requerimientos.	100%
Planeación del proyecto de software.	100%
Seguimiento del proyecto de software y sobrevista.	100%
Administración de subcontratos de software.	—
Aseguramiento de la calidad de software.	100%
Administración de la configuración de software.	100%
Enfoque al proceso de organización.	100%
Definición del proceso de organización.	100%
Programa de entrenamiento.	—
Administración de integración del software.	60%
Ingeniería de producto de software.	80%
Coordinación intergrupal.	—
Revisiones de pares.	—
Administración cuantitativa del proceso.	—
Administración de la calidad del software.	—
Prevención de defectos.	10%
Administración del cambio de tecnología.	—
Administración de cambio del proceso.	—

En la tabla se muestra como las técnicas utilizadas en el proceso empatan con las áreas clave del proceso (KPA's), en un gran porcentaje, aunque cabe mencionar que algunas de las KPA's no cubiertas se deben al tipo de proyecto y a que esta es una primera aproximación del modelo ajustado propuesto.

CONCLUSIONES

El caso presentado a través de los aspectos a reducir en esta primera aproximación a un modelo, ha permitido verificar su implementación sin problemas en la generación de este proyecto corto y en un equipo de desarrollo pequeño, habiendo obtenido como resultado en la aplicación del mismo las siguientes conclusiones:

* El proceso reducido pudo ser ejecutado sin problemas, es decir en relación a la situación planteada sobre resistencia al cambio por parte de la organización fue resuelta, esto se confirma por los desarrolladores experimentados mencionan que la aplicación de la metodología como tal es sencilla y fácil de implementar originando con esto una aceptación casi inmediata.
* El uso del PSP y TSP fue ejecutado a través de sus guías y tamplates (principales), logrando así planificar y controlar el proceso.
* A manera de medir la eficiencia del modelo, los objetivos de las KPA's descritas en la Tabla 2, fueron cubiertos casi en su totalidad por lo menos en las KPA's relacionadas a proyectos cortos.
* Los usuarios se han sentido satisfechos debido principalmente a su participación constante a lo largo del desarrollo de la aplicación, generando con esto entre otras cosas un sentido de pertenencia desde el inicio por parte de éstos.
* El esfuerzo del proyecto estimado inicialmente fue de total de 650 horas, la ejecución del proyecto real fue de 675 horas, logrando tener sólo 25 de horas de diferencia (la métrica fue muy asertiva). Las métricas aplicadas para medición del esfuerzo fue el COCOMO [20] considerando como parámetro una complejidad de proyecto simple ($PM = 2.4 (KDSI)^{1.05} X M$).
* Los desarrolladores generan especificaciones generales y detalladas del sistema que cumplen con las expectativas de los diferentes tipos de usuarios, y por lo tanto la disminución de modificaciones por falta de entendimiento o especificaciones incompletas o anómalas.
* El control sobre el proyecto es sencillo y por lo tanto los analistas, diseñadores y desarrolladores lo implementan sin dificultad en su interpretación.
* En relación a métricas sobre procesos queda pendiente aplicar el experimento en al menos un par de proyectos para medir su mejora basado en métricas de GQM (Goal-Question-Metric) [19].

REFERENCIAS

[1] Batista J., Dias de Figueiredo A. 2000. SPI in a Very Small Team: a Case with CMM. Software Process Improvement and Practice 2000; 5: 243-250.
[2] Basili, V.R., Rombach, H.D. 1998. The TAME project: towards improvement-oriented software environments. IEEE Trans. On Software Engineering, 14(6), 758-773. (Caps. 24, 25).
[3] Bohem, Barry., Using the Win Win Spiral Model: A case Study, Computer, Vol. 31, No. 07, Julio 1998.
[4] Brodman JG, Jonson DL. 1992. Software process rigors yield stress, efficiency. Signal Magazine, 55, August.
[5] Carpers, Jones, Critical Problems in Software Measurement, the IS Management Group, CA, 1993.
[6] Fugetta, A., Processo Software, Aspetti strategici e organizzativi, il Cardo editore in Venezia, 1994.
[7] Goth, G. "The Team Software Process: A Quiet Quality Revolution", IEEE Software, Vol. 17, No. 6, Noviembre/ Diciembre 2000, pp. 125-127.
[8] Hayes, W., Over, J.W., "The Personal Software Process (PSP): An Empirical Study of the Impact of PSP on individual Engineers", Reporte Técnico CMU/SEI-97-TR-001, ESC-TR-97-001, Dic. 1997.
[9] Horvart RV, Rosman I, Gyorkos J. 2000. Managing the complexity of SPI in small companies. Software Process - Improvement and Practice 5(1): 45-54.
[10] Humphrey, Watts S., Discipline for Software Engineering, Addison Wesley, 1995.
[11] Humphrey, Watts S., Introduction to the Team Software Process, Addison Wesley, 2000.
[12] Leung HK, Yuen TC. 2001. A Process Framework for Small Projects. Software process - Improvement and practice 6: 67 - 83.
[13] McAndrews, D. 2000. "The Team Software Process (TSP): An Overview and Preliminary Results of Using Disciplined Practice", Reporte Técnico CMU/SEI-2000-TR-015, ESC-TR.2000-015, Nov.
[14] Paul, M.C, Curtis, B., y Weber, C.V., Capability Maturity Model, Version 1.1., IEEE Software, vol. 10, No. 4, Julio 1993.
[15] Richardson I. 2001. Software Process Matrix: A small Company SPI Model. Software Process - Improvement and Practice 6(1): 157 - 165.
[16] Somerville, Ian. 2002. Ingeniería de Software. Addison Wesley.
[17] Dorling, A. SPICE, Consolidated product, part 1: Concepts and introductory guide. ISO/IEC, 1995.
[18] Rico, David F., PSP Personal Software Process, Executive Overview, SEI, 2000.
[19] Basili, V.R. y Green, S. Software Process improvement al the SEL. IEEE Software, 11 (4), 58-66. (Cap. 25). 1993.
[20] Boehm,B. y Royce, W. Ada COCOMO and the Ada Process Model. Proc. 5 th COCOMO User's Group Meeting, Pittsburgh, SEI (Cap. 23). 1989.
[21] Andre, Scholz, Performance Egineering Maturity Model, Magdeburgo University, Germany, 1999.