Modelo para la reducción del ciclo de desarrollo de software basado en CMM.

Francisco J. ALVAREZ.
Departamento de Sistemas Electrónicos, Universidad Autónoma de Aguascalientes.
Aguascalientes, Aguascalientes, C.P. 20138, México.

Alfredo WEITZENFELD.
Departamento de Ciencias Computacionales, Instituto Tecnológico Autónomo de México,
México, D.F.

RESUMEN.

El presente trabajo muestra un modelo basado en CMM (Capability Maturity Model) orientado en la reducción del ciclo de vida para su aplicación en proyectos y equipos de desarrollo pequeños. Se muestra el modelo general, así como los aspectos a considerar y las KPA's (Key Process Areas) cubiertas por el modelo. En la parte experimental el modelo fue aplicado a un producto de software (Sistema de Control Médico) obteniendo resultados interesantes como el hecho de apegarse al modelo de madurez mencionado y el incremento en el control del proyecto.

Palabras clave: Calidad de Software, Procesos de desarrollo, CMM (Capability Maturity Model), KPA's (Key Process Areas), Proyectos pequeños.

1. INTRODUCCIÓN.

Este trabajo presenta la solución al desarrollo de software en equipos pequeños, los cuáles pueden considerarse con un máximo de 20 personas [5]. La principal razón es que un gran número de organizaciones tienen estas características y prevalecen como empresas que aportan un gran potencial de productos y aplicaciones en la industria de software incluso en países dedicados a este tipo de industria como Irlanda [4][9].

Algunos de los problemas ha solucionar en esta industria es la disminución de los tiempos de desarrollo lo cual se ha logrado a través de los modelos de madurez de procesos [7], sin embargo los principales factores a los que se enfrenta con esta solución es el alto costo de implementación de modelos y la disminución de productividad en el período de aprendizaje de la organización [10].

Un beneficio adicional a la disminución del ciclo de desarrollo de software es que los pequeños equipos de construcción y proyectos cortos lograrían disminuir las fases y complejidad del proceso [6], estos ajustes a una flexibilidad de los pequeños proyectos y equipos de desarrollo lograrían el estandarizar procesos e incrementar la calidad de los productos.

Para adoptar el CMM en un mayor número de organizaciones y tamaños del proyectos, CMM presenta una serie de aspectos por resolver [3]:

1) La excesiva documentación del proceso de desarrollo de software, de la organización y de los aspectos relacionados.
2) La planeación, organización y dirección de KPA's orientadas a grandes organizaciones.
3) La amplitud de revisiones señaladas en las KPA's.
4) Los recursos limitados de las empresas que desarrollan software.
5) Los altos costos de entrenamiento.
6) Las prácticas sin relación al tipo de proyecto que apunta documentar CMM.
7) 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 grupo de desarrollo es de suma importancia para resultados favorables ya que altos niveles de habilidades y experiencia genera calidad en los productos [11], de aquí que el equipo no pueda 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).

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 (Key Process Areas) cubiertas de este nivel. La medición de la mejora del proceso de desarrollo, de otra forma implica una dificultad, ya que es complicado conocer qué medir [2].

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 [8]. 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:

- Disminución de las fases considerando las necesidades de un proyecto corto.
- La consideración de prioridad de actividades de equipos pequeños de desarrollo enfatizando en la construcción del software.
- Lograr los objetivos de las KPA's relacionados con la definición de procesos de proyectos cortos, es decir no aplicar KPA's que no sean necesarias en organizaciones pequeñas y proyectos cortos.
- 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.

2. MODELO PROPUESTO.

El modelo propuesto involucra el desarrollo de 12 actividades, estas actividades están conformadas como una estrategia para garantizar la implementación de CMM en la organización, así como obtener una mejora en el proceso de desarrollo:

1) Compromiso de la dirección en la consecución del proyecto.
2) Establecimiento del SPEG (Software Process Engineering Group).
3) Definición de meta de proyectos y organización.
4) Creación de una guía para el aseguramiento de las metas.
5) Plan de información sobre CMM en la organización.
6) Desarrollo de un proceso de software común.
7) Proceso trazado por CMM
8) Plan y operación de entrenamiento en la organización.
9) Colección de los datos y mejora del proceso.
10) Desarrollo del proceso pequeño (ajustado).
11) Una valoración de SQA (Software Quality Assurance), es decir el establecimiento del programa de aseguramiento de la calidad de software.
12) Mejora continua.

Ciclo De Desarrollo Reducido Para Proyectos Pequeños.

El ciclo de vida del proyecto mantiene una base a través de categorías y control de las 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 implementación con lo cual se obtiene la primera reducción del ciclo de desarrollo (ver *tabla 1*). Como el número de fases es reducido, también el esfuerzo gastado en seguimiento del proyecto. Puede reducirse la revisión de SQA y la colección de los datos necesarios para el seguimiento y control del proyecto.

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

Tabla 1. Fases en procesos pequeños y procesos estándar.

La *Figura 1* 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, es muy difícil esperar el entrenamiento antes de iniciar el desarrollo real de un proyecto.

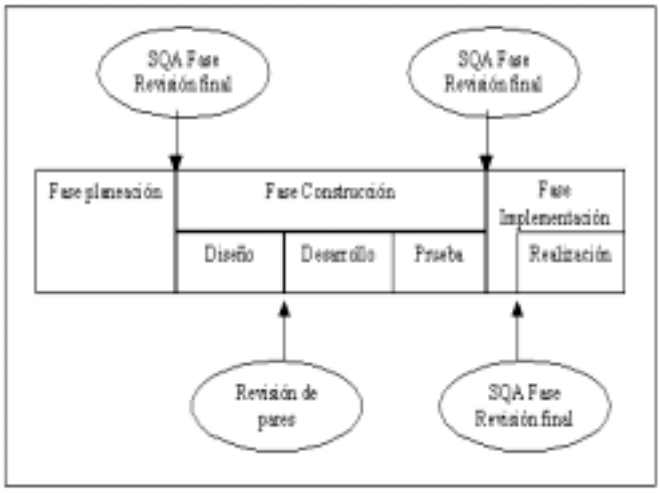

Figura 1. Revisión en el ciclo de vida de proyectos pequeños.

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 roles múltiples que se traslapan en un proyecto dado [10]. Hay restricciones básicas, en los papeles de SQA y miembros de equipos de prueba. Puesto que la función de SQA debe ser independiente del desarrollo del software que agrupa, el personal de SQA no debe involucrarse en el desarrollo. Un miembro del equipo de pruebas también puede ser un miembro del equipo de desarrollo de otro proyecto determinado.

3. RESULTADOS EN LA REDUCCIÓN DEL CICLO EN EL CASO DE ESTUDIO.

- Período desarrollo: Agosto 2003 – Diciembre 2003.
- Lugar: Estado de Aguascalientes, México.
- Características de los proyectos: Aplicación para la administración de registros de una colección de herbolaria, Sistema de control vehicular y Control de pendientes para empresa inmobiliaria. 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 tuvo que capacitar al equipo de desarrollo y monitorear la correcta aplicación del modelo reducido. La capacitación aproximadamente fue de 60 horas.

Disminución de fases.
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 [11], 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 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 alcanzadas en el producto formalmente con cliente.

5. *M*étricas. Se aplicaron “Function Points“ (Punctos Funcionales) al inicio del proyecto y final para verificar el cumplimiento de tiempos y tamaño de producto.

Obtención de los objetivos de KPA’s primordiales.
Se ha comparado el proceso de desarrollo reducido contra las 14 KPA’s del CMM. La selección de las KPA’s es la mas representativa para la obtención del nivel 2 (Repetible) y nivel 3 (Definido), así como la aplicación en proyectos cortos.

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. 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.

4. CONCLUSIONES.

El caso presentado a través de los aspectos a reducir en esta primera aproximación, ha permitido verificar su implementación sin problemas en la generación de este proyecto corto y un equipo de desarrollo pequeño.

El esfuerzo del proyecto estimado inicialmente fue de total de 650 horas, la ejecución del proyecto real fue de 675 horas, logrando tener solo 25 de horas de diferencia (la métrica fue muy asertiva) y los ingenieros de software generaron 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.

5. 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. (Cap. 24).

[3] Brodman JG, Jonson DL. 1992. Software process rigors yield stress, efficiency. Signal Magazine, 55, August.

[4] Center for Software Egineering. 1996. TRI-Spin Case Studies, Nos 1-16. Dublin, Ireland.

[5] Goldenson D, Herbesler J. 1995. After the appraisal: systematic survey of process improvement, its benefits, and factors that influence success. CMU / SEI-95-TR-009, Software Engineering Institute, Carnegie Mellon University, Pittsburg, Pennsylvania, USA.

[6] Horvart RV, Rosman I, Gyorkos J. 2000. Managing the complexity of SPI in small companies. Software Process – Improvement and Practice 5(1): 45-54.

[7] Humphrey, WS., 1990. Mnaging the Software Process. Addison Wesley Publishing.

[8] Jones GW.1990. Software Engineering. Wiley. [10] Kuvaja P, Messnarz R. 1996. BootStrap – a modern software process assessment and improvement methodology. Software Quality and Business Opportunities, Proceedings of the Tith European Conference on Software Quality, 16-20 th Sptember, Dublín, Ireland; 194-207.

[9] Kuvaja P, Messnarz R. 1996. BootStrap – a modern software process assessment and improvement methodology. Software Quality and Business Opportunities, Proceedings of the Tith European Conference on Software Quality, 16-20 th Sptember, Dublín, Ireland; 194-207.

[10] Leung HK, Yuen TC. 2001. A Process Framework for Small Projects. Software process – Improvement and practice 6: 67 - 83.

[11] Somerville, Ian. 2002. Ingeniería de Software. Addison Wesley.