Costos de la Calidad y del Mejoramiento del Proceso de Software

Mauricio Concha F. y Marcello Visconti Z.

Departamento de Informática

Universidad Técnica Federico Santa María

Valparaíso, Chile

e-mail: {mconcha,visconti}@inf.utfsm.cl

Resumen

Cuando en una organización de software se fijan objetivos de calidad, se debe definir un plan preciso que conduzca a la obtención de tales objetivos. El plan debe contemplar el mejoramiento de los procesos de software como forma fundamental de obtener calidad. Además, los efectos del plan deben ser cuantificados a través de conceptos específicos que componen el costo de la calidad. Al utilizar esta orientación, se puede señalar que el plan de mejoramiento debe ser costo-efectivo, con el objeto de lograr calidad y productividad para la organización. Este trabajo determina los conceptos claves para lograr calidad costo-efectiva, y de qué forma debe ser utilizado apropiadamente algún método de evaluación, como el modelo CMM o las normas ISO 9000, para desarrollar un plan de mejoramiento de procesos.

1.- Introducción.

El objetivo de la Ingeniería de Software es desarrollar un producto de software de alta calidad, a partir de recursos limitados, mediante un proceso en el cual se utilizan diversas metodologías, herramientas y estándares.

La calidad del producto de software obtenido en un proyecto no es un resultado aleatorio ni dependiente de eventualidades ni esfuerzos individuales, sino que la calidad obtenida estará determinada por la forma cómo se lleva a cabo el proceso de desarrollo. Este enfoque para lograr calidad pone el énfasis en el mejoramiento y perfeccionamiento contínuo del proceso de software.

La calidad de software está determinada por la calidad del proceso utilizado para desarrollar y mantener el software. Este concepto no es más que una extensión del principio que ha sido aplicado exitosamente en la industria de manufactura y de servicios en los últimos años (5). Esto indica que para la obtención de calidad en proyectos de software la atención debe ser puesta esencialmente en el proceso más que en el producto (9).

La productividad debe considerarse como un factor económico relevante cuando se define un plan de mejoramiento de procesos. Un plan bien definido y ejecutado de mejoramiento de procesos debe, por sí solo, asegurar calidad y aumentar productividad, basándose en la idea que los costos de producción son reducidos significativamente cuando se desarrolla software con un bajo índice de errores (3).

En este trabajo se analizan los factores que relacionan el aseguramiento de la calidad con el aumento de la productividad, y como éstos se reflejan en un plan de mejoramiento del proceso de software basado en el Capability Maturity Model (CMM) o en la certificación de calidad ISO-9000.

2.- Factores Económicos de la Calidad.

Cuando se fijan objetivos de calidad, se hace necesario que las organizaciones midan los resultados de sus esfuerzos destinados al mejoramiento de los procesos. Un enfoque adecuado para cuantificar los efectos de las iniciativas de mejoramiento es el enfoque basado en la idea de "costo de la calidad" de Phil Crosby (5). Extendiendo este enfoque, se pueden definir los costos de desarrollo como los costos de realización más los costos de la calidad. A partir de esto, se hace diferencia entre el costo de realizar algo correctamente la primera vez y el costo de rehacer el trabajo

Costos del Desarrollo = Costos de Realización + Costos de Calidad

En donde,

1.- Costos de Realización: Asociados con hacer las cosas correctas la primera vez. (Especificación de requerimientos, diseño, codificación).

2.- Costos de la Calidad:

2.1.- Costos de Prevención: Asociados a las actividades que intentan prevenir fallas antes que puedan llegar al producto.

2.2.- Costos de Evaluación: Asociados con revisiones e inspecciones de los productos generados durante el proceso de software.

2.3.- Costos de Corrección o de Rehacer el Trabajo (*rework*) debido a la ocurrencia de defectos.

Entre los costos de prevención se pueden destacar los costos de desarrollo de prototipos. Los prototipos son usados para lograr un claro entendimiento de requerimientos, funcionalidad o estructura del software a construir. El principal objetivo del desarrollo de prototipos es evitar incurrir

en errores, que en fases posteriores provocarían grandes costos de corrección (4).

Los costos de evaluación son asociados a esfuerzos destinados a revisiones e inspecciones formales.

Los costos que están asociados a la corrección de fallas y errores se clasifican como costos de *rework* o de rehacer trabajo, y usualmente están ocultos como trabajo real en muchos proyectos. Estos costos deben ser considerados como claves cuando se definen programas de mejoramiento de procesos.

Los costos de calidad son una herramienta de administración y proporcionan una visión de cómo la organización está funcionando. Si estos costos son altos, la organización está probablemente funcionando en forma ineficiente con ciclos de vida extensos, baja productividad y costos más altos de lo necesario. Por otro lado, bajos costos de calidad indican un proceso de desarrollo optimizado que se logra haciendo las cosas correctamente.

Es posible relacionar los costos de desarrollo con mediciones de tamaño de un producto, en puntos de función por ejemplo, con el fin de obtener datos de productividad para un ambiente de desarrollo. Según estos conceptos, obtenemos la siguiente relación:

Productividad: Tamaño del Producto / (Costos de Realización + Costos Calidad)

La siguiente relación define la variación de productividad:

$$\Delta P = \frac{T}{R + \Delta(r + p + e)}$$

En donde, P = Productividad.
R = Costos Realización.
r = Costos *rework*.

p = Costos prevención.
e = Costos evaluación.
T = Tamaño software

La variación de la productividad para un software de tamaño T estará dada principalmente por la variación del costo de la calidad (r+p+e).

Se considera el costo de realización (R) como constante o con una variación despreciable, debido a que a este costo no asocia los recursos asignados a detección y corrección de errores, principal razón de la variación de los costos de desarrollo de un proyecto.

A partir de esto, con el objeto de aumentar calidad y aumentar productividad se debe dar:

$$\Delta (r + p + e) <= 0$$

Con las siguientes condiciones:

$\Delta r < 0$, disminución costos de *rework*.

$\Delta p >= 0$, aumento costos de prevención.

$\Delta e >= 0$, aumento costos evaluación.

Por lo tanto, para obtener calidad costo-efectiva se debe cumplir estrictamente:

$$| \Delta r | >= | \Delta p | + | \Delta e |$$

Un sistema de calidad costo-efectivo conduce a incrementar la productividad y a una permanente reducción de costos, debido a que posibilita a la administración del proyecto reducir los costos de *rework* o de corrección de errores al colocar el énfasis en la prevención y en la revisión.

Los costos iniciales de establecer un sistema de calidad son compensados por los posteriores resultados logrados dentro de uno o dos años. Por ejemplo, *Raytheon Equipment Division* fue capaz de reducir los costos por fallas en un periodo de tres años desde 45% a 15%, y el costo de la calidad

se redujo al 20% de los costos del proyecto, todo esto logrado al mejorar sus procesos de software (5).

Es interesante señalar que en 1987, Barry Boehm estimó que eliminando el rehacer trabajo se podría reducir los costos del software entre 30% a 50% (8).

Reduciendo *Rework*.

El *rework*, que son las actividades asociadas con rehacer trabajo, posee varias componentes. De este modo es de valor analizar cada factor de esfuerzo o costo que contribuye principalmente a la reducción global de *rework* y tratar de determinar cuáles cambios en los procesos podrían lograr estas reducciones.

Además, importantes ahorros en costos de *rework* se pueden lograr a partir de aumentos leves en el costo de otros procesos. Por ejemplo, en el caso de la compañía *Raytheon* en EE.UU. (5), en donde se desarrolló un completo plan de mejoramiento de su proceso de software, los costos de diseño y codificación se elevaron levemente debido a que las inspecciones formales reemplazaron a las revisiones informales. Esta acción de cambio, llevó a ahorros en *rework* debido a que permitió descubrir problemas en el código antes de la fase de integración de software, eliminando de esta manera el *retesting*.

Con el objeto de reducir el *rework* se debe conocer cuáles son las actividades que contribuyen más fuertemente a los costos de éste, es decir, qué tareas dentro del desarrollo se deben volver a hacer debido a defectos.

Detección y Corrección Temprana de Defectos.

El número de defectos puede ser reducido sustancialmente al adoptar una estrategia para prevención de defectos, pero éstos no serán eliminados completamente. Por esta razón, se debe

detectar y corregir los defectos tan pronto como son introducidos en el desarrollo. Mientras más tiempo transcurre y un defecto no es detectado, más caro será corregirlo. Algunos datos disponibles señalan que un defecto de diseño puede llegar a ser 15 veces más caro corregirlo en la fase de *testing* que en la fase de diseño (7).

Inspecciones Formales.

A través de la revisión de los factores presentados, se puede señalar la importancia de las inspecciones para lograr reducir los costos de *rework*, basándose en que la detección y corrección de defectos debe hacerse en la misma fase en que suceden. Como ejemplo, en Hewlett-Packard (HP) se estimó que aproximadamente un tercio de todos los costos de software son provenientes de *rework*, y las inspecciones pueden ahorrar el 60% de tales costos (4).

Testing.

El principal factor que afecta al *testing* es el número de defectos en el producto al comenzar esta fase. Ante presiones por cumplimiento de plazos y conociendo que se ha desarrollado un producto aún con defectos, se puede optar por prolongar los tiempos de entrega o liberar un producto de baja calidad. La solución económica a este problema es lograr remover la mayor cantidad de defectos antes de comenzar el *testing* (6). Es deseable encontrar y corregir los defectos antes que se comience el *testing*, pero eliminar todos los defectos es imposible, por lo que esta fase de *testing* requiere una importante planificación con el objeto de lograr un software de calidad.

3.- Plan de mejoramiento de procesos.

Para lograr entender cuáles mejoras son más urgentes para una organización, debe comprenderse inicialmente las diferencias entre una organización inmadura y una organización madura.

Una organización inmadura no posee una base objetiva para evaluar la calidad del producto. Existe un pequeño entendimiento de cómo las fases del proceso de software afectan a la calidad, y por lo tanto la calidad del producto es difícil de predecir. Más aún, las actividades que pretenden mejorar la calidad, como revisiones o *testing*, son a menudo eliminadas cuando los proyectos están generalmente fuera de plazos. En este tipo de organizaciones se desarrolla con altos riesgos, baja productividad, altos costos y calidad incierta.

Una organización madura posee la habilidad para administrar el desarrollo y mantención de software. Los administradores se preocupan de la calidad del producto y de los procesos. Existe una base objetiva para evaluar la calidad y analizar problemas. Plazos y presupuestos se basan en estimaciones correctamente realizadas sobre datos históricos. En este tipo de organizaciones se desarrolla con bajo riesgo, alta productividad, dentro de presupuestos y con aseguramiento de calidad (2,10).

El camino en el mejoramiento de procesos debe estar dado desde la inmadurez hacia un mayor grado de madurez, en un proceso de mejoramiento contínuo. Al revisar la relación entre la productividad y los costos de la calidad, se puede generalizar y determinar tres fases en el mejoramiento de los procesos de software por los cuales debería evolucionar una organización:

FASE 1	Costos de calidad altos dados por un gran costo en rehacer trabajo y en costos de evaluación dados por *testing*. Se obtiene baja productividad y calidad incierta.
FASE 2	Se mantiene el costo de calidad desde el estado anterior, pero la relacion entre los costos de prevención, evaluación y de rehacer el trabajo cambian. Se disminuye el costo de rehacer el trabajo y se aumentan los costos de prevención y de evaluación, con lo que se logra un producto de mayor calidad y se eliminan riesgos; la productividad no disminuye pero aumenta la calidad del proceso de software.

FASE 3	Disminución de los costos de calidad, dado por el ahorro de recursos en rehacer el trabajo. Se mantiene los costos de prevención y evaluación con lo que se asegura calidad en el producto final. La productividad aumenta y se asegura la calidad..

A partir de esto, una organización inmadura que intenta asegurar la calidad corre el riesgo de que aumente los costos del proyecto, y por consecuencia disminuya la productividad. El éxito de una organización se logra cuando la obtención de calidad va junto al aumento de la productividad, y esto se observa en donde el rehacer trabajo o *rework* ha sido reducido casi completamente. Los efectos del mejoramiento progresivo de los procesos de software versus los costos de desarrollo se ven graficados en la Figura 1.

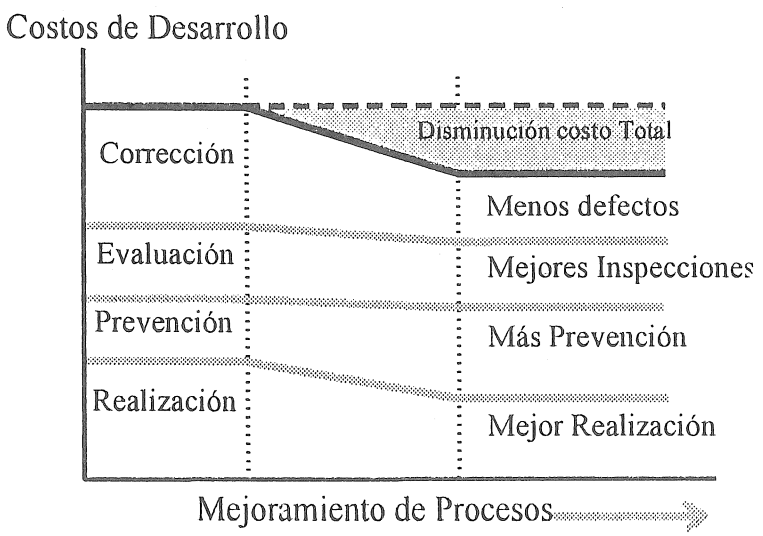

Figura 1.- Relación entre los costos de desarróllo durante el mejoramiento de procesos.

Evaluación de Procesos de Software

Existen esencialmente dos razones por las cuales las organizaciones que desarrollan proyectos de software efectúan alguna evaluación de sus procesos de desarrollo, basándose en estándares que

sean objetivos y ampliamente aceptados :

1) Obtener ventajas competitivas

2) Definir objetivos estándares para elaborar un plan de mejoramiento de procesos.

Sin embargo, el principal objetivo de la evaluación de los procesos de software en una organización debe apuntar hacia obtener un sistema de calidad costo-efectivo. Este objetivo debe ser la base de un plan de mejoramiento de procesos que se oriente en los factores económicos de la calidad (costo de la calidad, productividad).

Al detallar un plan en actividades se debe tomar como orientación algún estándar ampliamente reconocido, como por ejemplo ISO 9000 o el *Capability Maturity Model* (CMM), el cual preste una evaluación objetiva y ampliamente aceptada a los logros obtenidos en el desarrollo del plan.

Cuando se ha definido cómo lograr un sistema de calidad costo-efectivo, entonces se puede incorporar actividades específicas con el fin de lograr la certificación deseada. La Figura 2 ilustra esta relación.

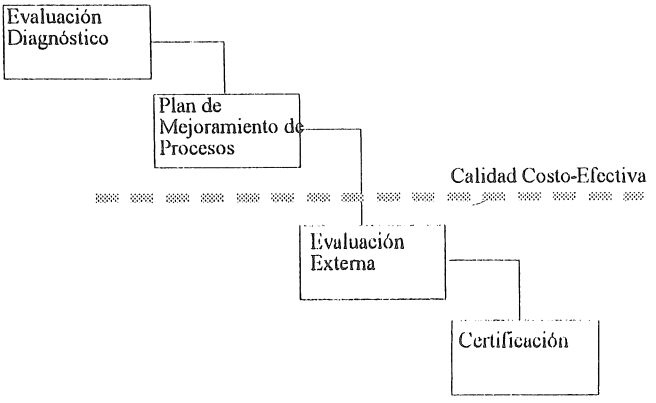

Figura 2.- Fases para lograr Calidad Costo-Efectiva y luego Certificación.

Normas ISO-9000 y Modelo CMM.

El *Capability Maturity Model* desarrollado por el *Software Engineering Institute* (SEI), y la serie de estándares ISO 9000 desarrollado por la *International Organization for Standarization* (ISO), son estándares de evaluación y certificación que poseen un interés común en la calidad y en la administración de procesos de software (10).

El estándar específico en la serie ISO 9000 que interesa cumplir a las organizaciones que desarrollan software es ISO 9001, que define normas para las etapas de diseño, desarrollo, producción, instalación y servicio. Como ayuda ISO proporciona una guía específica para la aplicación de ISO 9001 al desarrollo de software denominada ISO 9000-3.

El CMM define un marco de referencia para la madurez de los procesos de software y define 5 niveles, en los cuales cada nivel proporciona una base para lograr los niveles superiores.
Los conceptos que se manejan en ambos estándares son similares, pero la diferencia entre ellos radica en su filosofía fundamental. ISO 9001 identifica los requerimientos mínimos para un sistema de calidad, mientras CMM se basa en la necesidad de un contínuo mejoramiento de procesos a través de la evaluación de la madurez de los procesos (6).

Por dicha razón, es posible señalar que ISO 9001 conduce en forma implícita a un mejoramiento de los procesos a través de contínuas correcciones con el fin de buscar la certificación, mientras CMM es explícito en su objetivo del contínuo mejoramiento de procesos.

Lo esencial para una organización es enfocar sus esfuerzos de mejoramiento en lograr un verdadero sistema de calidad costo-efectivo, en vez de solo esforzarse en lograr una marca, ya sea ésta un nivel de madurez o una certificación.

Existen esfuerzos por parte de ISO para combinar características de CMM e ISO 9000 y otros

métodos como *Bootstrap* de ESPRIT, con el fin de desarrollar un conjunto de estándares denominado SPICE (*Software Process Improvement and Capability dEtermination*). El estándar SPICE se orienta principalmente en el proceso de software, pero se extenderá también a temas como recursos humanos, tecnología, prácticas de gestión, apoyo al cliente. SPICE es desarrollado bajo ISO/IEC/JTC1/SC7, cuerpo responsable de estándares en Ingeniería de Software , en contraste con ISO 9000 que fue desarrollado por ISO TC176, el cual es responsable de estándares para sistemas de administración de calidad en general (10).

Definición de un Plan.

Cuando se desarrolla un plan de calidad orientado por un estándar, si éste es implementado en forma errónea puede ser más negativo que positivo para la organización.

Es posible caer en prácticas que solo conduzcan a llenarse de papeles y de definiciones extensas de tareas. Para evitar esto, la atención debe ser puesta en el centro del problema, es decir, aplicar los conceptos del estándar como una guía para desarrollar software sin defectos desde las primeras fases del proceso de software.

4.- Conclusiones y Proyecciones Futuras.

Ante la exigencia de un mercado cada día más competitivo, se hace necesario producir software de alta calidad. Con este objetivo una organización debe desarrollar un plan que conduzca a la obtención de software de calidad, cuya orientación debe estar puesta en el mejoramiento contínuo de los procesos de desarrollo de software y debe basarse en estándares de evaluación que sean objetivos y ampliamente aceptados.

Un plan de mejoramiento basado en un estándar ya sea ISO 9000, CMM u otro, debe ser costo efectivo en primer lugar, para así lograr convertir a la organización en una organización madura y

lograr de este modo los niveles de calidad y productividad esperados.

La obtención de metas de calidad significará un costo, al cual se ha denominado costo de la calidad, que define un enfoque adecuado para cuantificar los efectos de las iniciativas de mejoramiento considerando el factor económico de la calidad.

¿ Cómo lograr calidad costo-efectiva ?. La respuesta está dada en la descomposición de los costos de la calidad y en la determinación de los factores que pueden influir significativamente en tales costos y que pueden ser evitados. El factor preponderante corresponde al *rework* o rehacer trabajo al corregir errores detectados.

El plan organizacional debe orientarse a lograr calidad costo-efectiva. Para conseguir ese objetivo debe reducir principalmente los costos de *rework*, lo que implica incorporar revisiones e inspecciones formales a las fases del desarrollo, evitando de este modo que los defectos permanezcan en el proceso y deban ser corregidos en forma posterior, con el conocido aumento de los costos por detección y corrección tardía de defectos. Estos conceptos deben estar presentes en la definición del plan de mejoramiento.

Es importante considerar que el logro de una certificación no es un fin en sí mismo, sino que debe ser utilizado como una orientación para lograr el cumplimiento del principal objetivo: el mejoramiento de procesos, basándose en un plan que considere conceptos de costo de calidad y de productividad. Esto con el fin de obtener un proceso de software que sea realmente costo-efectivo.

Se consideran a futuro estudios referentes al desarrollo de un modelo de plan de mejoramiento de procesos, basado en estándares de evaluación y certificación, que sea aplicado a organizaciones inmaduras, y que a través de un conjunto de métricas pueda ser evaluado en las distintas fases de la evolución de los procesos del software. Además, es necesario continuar el estudio en lo referente a

la priorización dentro de la Ingeniería de Software de los conceptos de "costo de calidad" , "calidad costo-efectiva" y "evaluación y certificación de calidad".

5. - Referencias.

1.- Norma Chilena Nch-ISO 9000-3 1994

2.- M. Paulk y B.Curtis, "Capability Maturity Model, version 1.1"
IEEE Software, pág. 18-26, Julio 1993.

3.- M. Rosch, "Getting the Best Out of ISO 9000"
Objects in Europe, pág. 13-15, Verano 1994.

4.- W. Humphrey, T. Snyder, "Software Process Improvement at Hughes Aircraft"
IEEE Software, pág. 11-23, Julio 1991.

5.- R. Dion, "Process Improvemenet and the Corporate Balance Sheet"
IEEE Software, pág. 28-35, Julio 1993.

6.- W. Humphrey, "A discipline for software Engineering"
Addison Wesley Publishing Company, 1995.

7.- E. Bryan, "CP-6: Quality and Productivity measures in the 15 year life cycle of an Operating System", Software Quality Journal, pág. 129-144, 1993.

8.- B. Boehm, " Understanding and Controlling Software Costs"
IEEE Transactions on Software Engineering", pág. 1462-1475, Octubre 1987.

9.- J. Gray, " Quality Costs: A Report Card on Business"
Quality Progress, Abril 1995.

10.- CMU/SEI, "The Capability Maturity Model, Guidelines for improving the software process". Addison Wesley Publishing Company, Inc. 1995.