REVISTA COLOMBIANA DE COMPUTACIÓN
Volumen 2, número 1
Págs. 75-87

Estimación del Coste de la Calidad del Software a través de la Simulación del Proceso de Desarrollo

Mercedes Ruiz Carreira[1] Isabel Ramos Román[2]

Resumen

El objetivo principal del control de calidad no es otro que detectar y corregir los errores que surgen durante el proceso de desarrollo de un producto software, así como garantizar que el producto final responde a las expectativas del cliente. Las actividades de control de calidad suponen un coste añadido al Proyecto de Desarrollo de Software (PDS), coste que está plenamente justificado pero que, en la mayoría de los casos, es el primero que sufre recortes cuando la presión del presupuesto y/o del tiempo aumenta de forma inesperada. En muchas ocasiones, la estimación del coste de las actividades de calidad para un proyecto se realiza como una parte porcentual del presupuesto total estimado para la ejecución del mismo. De esta manera, resulta imposible determinar cuál ha sido la eficiencia de dichas actividades sobre el producto desarrollado y cómo podrían haberse mejorado, si no resuelto, los problemas surgidos durante el desarrollo en torno a la calidad del producto. En este trabajo aportamos un nuevo enfoque para medir la eficiencia, en el ámbito de coste y rendimiento, de las actividades de aseguramiento de la calidad sobre un PDS. Para ello, proponemos combinar la simulación de modelos dinámicos, como alternativa a la costosa y, en la mayor parte de las ocasiones imposible, experimentación en la gestión de proyectos, con el método CoSQ.

Palabras claves: *Gestión de proyectos software, coste de la calidad del software, modelos dinámicos de procesos, simulador de proyectos.*

Abstract

Some of the main objectives of quality assurance practices are to detect and correct the errors that arise during the development of a software product, as well as to guarantee that the final product conforms to the client's requirements. Quality initiatives suppose an added cost to the software development projects, cost that is totally justified but which, in most of the cases, is the first one that is cut when the pressure of the budget and/or the time unexpectedly increases. In many occasions, the cost estimation of the quality initiatives consists of a percentage of part of the total budget considered for project development. This way, it is impossible to determine which the efficiency of those activities has been and how the quality problems arisen during the development could have been improved, if not solved. In this work we contribute a new approach to measure the efficiency, in the scope of cost and yield, of the software assurance activities.

[1] Dpto. de Lenguajes y Sistemas Informáticos. Universidad de Cádiz. Escuela Superior de Ingeniería. C/ Chile, nº1 11003 – Cádiz (Spain) e-mail: mercedes.ruiz@uca.es
[2] Dpto. de Lenguajes y Sistemas Informáticos. Universidad de Sevilla Facultad de Informática y Estadística. Avda. Reina Mercedes, s/n 41012 – Sevilla (Spain) e-mail: isabel.ramos@lsi.us.es

We propose to combine the simulation of dynamic models, as an alternative to the expensive, and in most of the occasions impossible, experimentation in the project management field, with the CoSQ method.

Keywords: *Software project management, cost of software quality assurance, dynamic process modelling, project simulator.*

1 Principios del Coste de la Calidad del Software (CoSQ)

El coste de la calidad, CoQ, (Cost of Quality) es una técnica introducida por Juran [Juran 1996] con el fin de proporcionar a los directores de proyectos un instrumento que les permita justificar la promoción de mejoras en el proceso de desarrollo. [Crosby 1979] aportó otra definición más acorde con el uso inicial de la técnica: "captar la atención de los directores sobre los aspectos relativos a la calidad y proporcionar una base que permita comprender y evaluar cómo se mejora la calidad cuando se introducen nuevas estrategias". Sin embargo, el uso de CoQ se ha extendido rápidamente más allá de su propósito inicial de obtener información sobre la mejora de la calidad. En la actualidad, se emplea en la mayoría de las industrias de manufacturas y de servicios, no sólo para obtener información sobre la calidad del producto, sino también para controlar los costes de las propias actividades de calidad e identificar los escenarios en los que es posible aplicar una reducción de estos costes sin comprometer la calidad del producto final.
CoQ se basa en medir dos tipos principales de costes: los costes originados por la pérdida de calidad en el producto y los costes debidos al propio proceso de alcanzar la calidad en el proyecto. Los costes motivados por la pérdida de calidad se dividen a su vez en costes debidos a fallos internos y costes debidos a fallos externos. Los costes derivados del proceso de alcance de la calidad se dividen en costes de prevención y costes de valoración. La Tabla 1 muestra una definición y ejemplos de estos costes. Resulta obvio descubrir que los costes para alcanzar la calidad y los costes debidos a la pérdida de calidad están relacionados de forma inversa: conforme la inversión dedicada a la mejora de la calidad aumenta, los costes originados por pérdida de calidad disminuyen. Esta relación y su efecto sobre el coste total de la calidad, TCoQ, (Total Cost of Quality) se muestran normalmente mediante un conjunto de dos curvas bidimensionales que representan los costes frente a una medida de calidad.

Categoría	*Definición*	*Costes típicos del software*
Errores Internos	Fallos de calidad detectados antes de la entrega al cliente	Gestión de errores, corrección, repetición de pruebas
Errores Externos	Fallos de calidad detectados después de la entrega al cliente	Soporte técnico, investigación de las quejas, notificación de errores por el cliente
Valoración	Descubrimiento de la condición del producto	Pruebas y actividades asociadas. Auditorías de control de calidad
Prevención	Esfuerzo para asegurar la calidad del producto	Administración de QA, inspecciones, mejoras al proceso

Tabla 1: Definición de las categorías de costes de la calidad

Mientras que los costes de las actividades de aseguramiento de calidad y de la mejora del proceso han sido objeto de mucha investigación durante más de 20 años [Alberts 1976], no hay una progresión similar en la determinación del coste de la calidad del software (CoSQ) [Hersh 1993] y [Knox 1993]. El modelo de Knox y los estudios de Raytheon [Dion 1993] constituyen notables excepciones ya que sí emplean de forma explícita el coste de la calidad del software. Los trabajos de Knox

demostraron uno de los beneficios de medir y utilizar CoSQ que no es otro que la necesaria justificación para la inversión en iniciativas de calidad. Por otro lado, [Plunkett 1990] señalaron que la utilización del CoSQ proporciona otros muchos beneficios de entre los que destacan:

- CoSQ puede utilizarse para comparar mejoras en los procesos e identificar la más efectiva. Por tanto, CoSQ proporciona las bases para medir y comparar la efectividad de los costes de todas las mejoras acometidas en la organización.
- CoSQ permite identificar las distintas mejoras de calidad candidatas para una organización. Una observación de los componentes de CoSQ revela los costes que son más elevados en un área concreta. Por ejemplo, unos costes de valoración elevados en las pruebas de integración pueden indicar que no se tuvieron en cuenta correctamente las interfaces durante el diseño, o que se han realizado cambios de última hora en el mismo. El análisis de las causas de estos costes puede proporcionarnos las bases para las iniciativas de calidad necesarias en el proceso de desarrollo.
- CoSQ proporciona una medida para comparar el éxito de distintos proyectos. Incluso dentro de una misma organización, el proceso de software puede variar ampliamente de un proyecto a otro. Los muchos factores, tangibles e intangibles, que caracterizan un proyecto hacen muy difícil compararlos. CoSQ puede medirse para cada proyecto y luego poder compararlos sin problema.
- CoSQ ofrece una base para estimar el coste de la operación de calidad. En vez de asignar a las actividades de calidad una parte porcentual del presupuesto total, la dirección puede estimar cuál será el gasto de las operaciones de calidad en función de los objetivos fijados.

Cada uno de los procesos de desarrollo de software puede ser sometido a muchas y diferentes mejoras, de forma que en el proceso de decisión entre todas ellas, resultaría muy deseable conocer los efectos de una mejora concreta sobre el proceso antes de su implementación. Estos efectos pueden preverse a través de la experiencia de otros proyectos u organizaciones (en estudios publicados) o a través de estudios pilotos. Sin embargo los procesos de software varían significativamente de una a otra organización y los estudios pilotos pueden ser costosos de implementar y difíciles de controlar para el estudio de una variable.

Una alternativa a estos problemas es el empleo de la simulación de los procesos software. El análisis de la gestión de los proyectos software a través de la simulación de sus procesos es una técnica que está despertando un gran interés investigador en la actualidad. Los estudios realizados por Abdel-Hamid y Madnick, publicados en [Abdel-Hamid 1991] ofrecieron una nueva perspectiva para estudiar los PDS utilizando la teoría de dinámica de sistemas. Esta teoría se define como la aplicación de los principios y técnicas de control para sistemas realimentados, al modelado, análisis y comprensión del comportamiento dinámico de sistemas complejos. El principio fundamental del modelado dinámico es la relación causa-efecto y la potencialidad que se alcanza cuando múltiples relaciones de este tipo se conectan formando una relación circular llamada lazo o bucle de realimentación. Los modelos dinámicos de PDS recogen las relaciones causales que se producen en el interior del proyecto y en las cuales intervienen factores relativos al personal técnico, al propio producto y al proceso de desarrollo.

Un modelo dinámico está formalizado matemáticamente por un sistema de ecuaciones diferenciales que recogen las restricciones existentes entre magnitudes que evolucionan en el tiempo.

En este estudio empleamos un modelo dinámico de simulación de proyectos software que hemos desarrollado y que denominamos Modelo Dinámico Básico (MDB) [Ruiz 2000]. Este modelo se ha elaborado, principalmente, aplicando la teoría de simplificación de modelos de [Eberlein 1989] al modelo original de [Abdel-Hamid 1991]. Las ecuaciones de este modelo básico, han sido adaptadas a un entorno de simulación gráfico como Vensim® y constituyen el núcleo de un simulador de PDS que nos permite simular diferentes políticas de gestión y evaluar sus efectos de forma interactiva.

2 Subsistemas de Aseguramiento de Calidad y Pruebas del Modelo Dinámico para PDS

El MDB integra las funciones típicas de gestión, planificación, control y personal, así como las actividades propias de la producción de software (diseño, codificación, pruebas, etc.). Resulta adecuado para simular proyectos de tamaño medio (15 – 100 KLDC) con requisitos estables a lo largo de todo el proyecto. Para poder simular las funciones anteriormente indicadas, el modelo se compone de cuatro subsistemas comunicados entre sí por medio de variables auxiliares. Estos subsistemas se representan gráficamente en la Figura 1 y se describen brevemente a continuación:

a) Subsistema de Gestión de Recursos Humanos. Contiene los aspectos relativos a las actividades de contratación, despido, formación y rotación del personal técnico del proyecto.

b) Subsistema de Producción del Software. Contiene, a su vez a otros cuatro subsistemas:

- Asignación de esfuerzo. Se encarga de distribuir el esfuerzo del personal técnico entre las distintas actividades del proyecto (diseño, codificación, aseguramiento de calidad y pruebas).
- Desarrollo. Contiene las actividades de diseño y codificación.
- Aseguramiento de calidad. Encargado de las actividades de detección y corrección de errores.
- Pruebas del sistema. Su misión consiste en asegurar que todos los elementos funcionen conjunta y correctamente, así como que el producto satisfaga los requisitos iniciales del cliente.

c) Subsistema de Control. Se encarga de medir el estado actual del proyecto (progreso percibido, consumo de recursos, etc.) e informar al subsistema de planificación.

d) Subsistema de Planificación. Con las estimaciones iniciales del proyecto y la información recibida del subsistema de control, se encarga de tomar las acciones correctivas necesarias.

Una vez vistos los componentes principales del MDB nos centraremos en estudiar el funcionamiento de los subsistemas que modelan el comportamiento de las actividades de calidad y de pruebas del sistema.

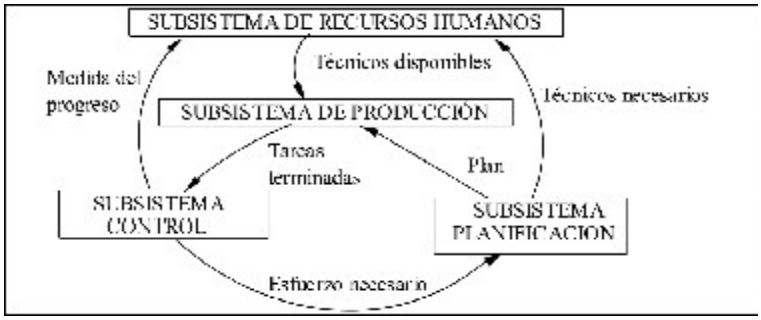


Figura 1: Subsistemas principales del MDB

2.1 Subsistema de Aseguramiento de Calidad

Como hemos dicho, el subsistema de control de calidad es el encargado de detectar y corregir los fallos a medida que éstos se producen en el proceso de desarrollo. En un proyecto software existen dos conjuntos de factores que afectan a la propia generación de errores y a la capacidad de detección que alcance el personal técnico. El primer conjunto de factores está constituido por las característi-

cas que son propias de la organización que desarrolla el proyecto (empleo de técnicas estructuradas, calidad del equipo técnico, etc.) y el segundo conjunto, contiene los aspectos que son dependientes del proyecto en sí mismo (complejidad, lenguajes empleados, tamaño, etc.). Estos dos conjuntos de factores pueden variar de una organización a otra y también entre proyectos de una misma organización, pero sí es claro que para un mismo proyecto son estables. El efecto de estos dos conjuntos de factores determina lo que denominamos, la capacidad nominal de generación de errores en un proyecto dado. Por otro lado, el volumen de errores que pueden producirse en un proyecto también está influenciado por otros factores de naturaleza dinámica como son la presión del plazo (cuanto mayor es la presión con la que se trabaja mayor será el número de errores que se cometan) y la experiencia del personal (la falta de experiencia en un técnico no sólo disminuye su productividad, sino que aumenta el número de errores que comete en sus actividades). Estos últimos factores de naturaleza dinámica, se unen a los de naturaleza estática constituyendo la capacidad real de generación de errores en un proyecto.

Por otro lado, la capacidad de detectar los errores del proyecto viene determinada por el esfuerzo asignado a las actividades de aseguramiento de calidad, por la experiencia del personal técnico y por el esfuerzo necesario para corregir un error. Consideraremos que los errores de diseño son 1,6 veces más costosos de detectar los errores de codificación según [Boehm 1981]. La asignación de esfuerzo a las actividades de calidad se puede realizar de dos formas diferentes. La primera de ellas consiste en asignar un valor fijo para todo el ciclo de vida que procede de aplicar un porcentaje determinado al esfuerzo disponible para las actividades de desarrollo. La alternativa a esta asignación fija de esfuerzo viene representada por asignar un esfuerzo que va a ser variable a lo largo del ciclo de vida y que dependerá fundamentalmente de la situación de presión de plazo en la que se encuentre el proyecto en cada momento.
Además, el modelo también contempla la situación real de que hay errores que son más difíciles de detectar que otros y que algunos errores pasarán por las actividades de aseguramiento de calidad sin ser descubiertos. El número de errores corregidos en el proyecto es una función del esfuerzo asignado a las actividades de corrección y del esfuerzo necesario para corregir un error. En el cálculo de esta variable, se vuelven a tener en cuenta aspectos tales como el tipo de error (la corrección de un error de diseño resulta 1,5 veces más costosa que la corrección de un error de codificación [Boehm 1981]). Es importante destacar que las ecuaciones del modelo también reflejan la posibilidad de que el propio proceso de corrección introduzca errores en el software que se está desarrollando. Este control se realiza a través de una variable que mide el porcentaje de malas correcciones.

La Figura 2 muestra el diagrama causal de este subsistema mientras que la Tabla 2 recoge una breve descripción de las variables empleadas.

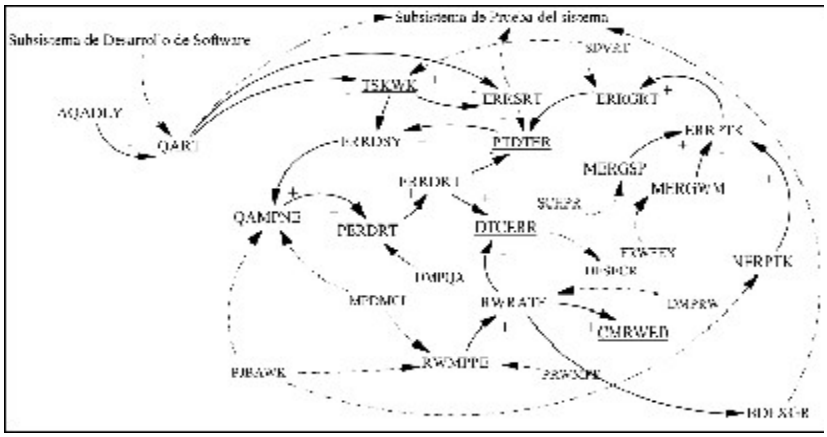


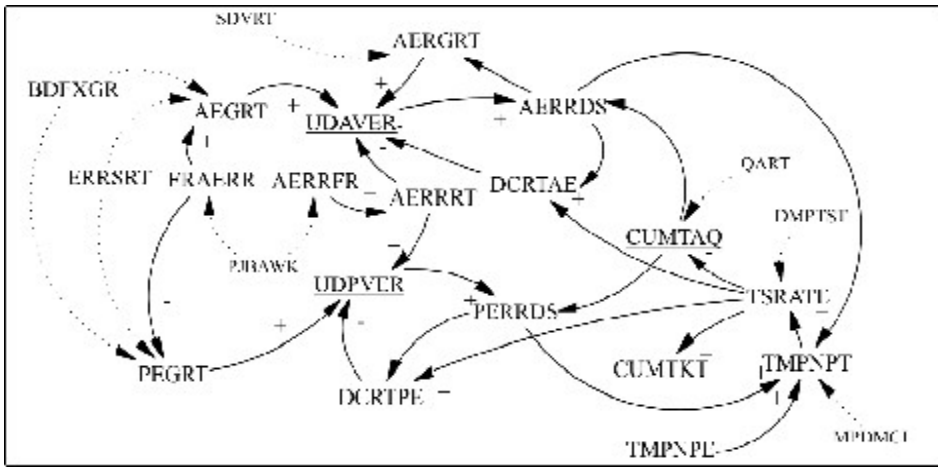
Figura 2: Diagrama causal del Subsistema de Aseguramiento de Calidad

SDVRT	*Flujo de tareas desarrolladas*	PTDTER	*Errores potencialmente detectables*
ERRDRT	*Flujo de detección de errores*	PERDRT	*Flujo potencial de detección de errores*
DMPQA	*Esfuerzo diario asignado a SQA*	QAMPNE	*Esfuerzo de SQA necesario para detectar un error*
DTCERR	*Errores detectados*	DESECR	*Flujo de corrección de errores deseado*
RWRATE	*Flujo de corrección de errores*	MPDMUL	*Multipl. debido pérdidas motivación/comunicación*
DMPRW	*Esfuerzo diario asignado a corrección*	BDTXGR	*Flujo genera. errores por correcciones defectuosas*
CMRWED	*Errores corregidos*	RWMPPE	*Esfuerzo actual necesario de corrección por error*
PJBAWK	*Trabajo realizado*	PRWMPE	*Esfuerzo de corrección por error estimado necesario*
ERRGRT	*Flujo de generación de errores*	ERRPTK	*Número de errores por tarea*
NTRPTK	*Número nominal de errores por tarea*	MFRGWM	*Multip. genera errores debido a la nueva tecnica*
SCHPR	*Presión*	MERGSP	*Multip. generación de errores debido a la presión*
FRWFEX	*Porcentaje de técnicos con experiencia*	ERRDSY	*Densidad de errores*
TSKWK	*Nivel de tareas para revisar y corregir*	ERRSRT	*Flujo de errores no detectados*
AQADLY	*Retra. media en las actividades de SQA*	QART	*Flujo de realización de las actividades de SQA*

Tabla 2: Nombre y breve descripción de las variables del Subsistema de Aseguramiento de Calidad

2.2 Subsistema de Pruebas

Los errores no detectados en las actividades de Aseguramiento de Calidad o los debidos a correcciones erróneas se deben detectar en las actividades de pruebas. Se considera que a esta fase pueden llegar dos grandes tipos de errores: los errores activos, que provienen de un error de diseño no detectado y que probablemente se han reproducido como errores de codificación; y errores pasivos, que no son más que errores de codificación no detectados. El proceso de propagación y reproducción de los errores activos así como el coste de detectarlos y corregirlos se recoge en el modelo. La capacidad de probar tareas viene determinada por el esfuerzo asignado a las actividades de pruebas, el esfuerzo requerido para probar una tarea y la eficiencia del personal técnico asignado. Las actividades de prueba finalizan cuando todas las tareas desarrolladas han sido probadas dándose por concluido, en ese momento, el proyecto, ya que el modelo no contempla la fase de mantenimiento.



La Figura 3: Diagrama Causal del Subsistema de pruebas y Tabla 3 describe de las variables que aparecen en él.

SDVRT	*Flujo de tareas desarrolladas*	QART	*Flujo de realización de las actividades de SQA*
FIRAWK	*Trabajo realizado*	RDFXER	*Flujo generac. errores por correcciones defectuosas*
ERRSRT	*Flujo de errores no detectados*	AERRDS	*Densidad de errores activos*
TSRATE	*Flujo realización actividades de Prueba*	CUMTQA	*Número total de tareas revisadas y corregidas*
AERGRT	*Flujo de regeneración de errores activos*	UDAVER	*Errores activos no detectados*
UDPVER	*Errores pasivos no detectados*	DCRTPF	*Flujo detección y corrección de los errores pasivos*
AERRRT	*Flujo de desactivación de errores activos*	AERRFR	*Porcentaje diario eliminación errores activos*
PEGRT	*Flujo de generación de errores pasivos*	CUMTRT	*Número total de tareas probadas*
AEGRT	*Flujo de generación de errores activos*	DCRTAE	*Flujo de detección/activación de los errores activos*
PERRDS	*Densidad de errores pasivos*	UMPTST	*Esfuerzo diario para Pruebas*
PRAERR	*Porcent. de errores activos no detectados*	MPDMCL	*Multipl. debido a pérdidas motivación/comunicación*
TVPNPF	*Esfuerzo necesario para Prueba por error*	TVPNPT	*Esfuerzo necesario en Prueba por tarea*

Tabla 3: Nombre y breve descripción de las variables del Subsistema de Pruebas

3 Determinación del CoSQ de un PDS simple

Para este estudio se calibró el MDB con el objeto de reproducir un proyecto de tamaño medio (24 KLDC). Para ello, se inicializaron las variables y parámetros del modelo con los valores mostrados en la Tabla 4.

La utilización de estos valores se justifica en la disposición de abundante información sobre este proyecto y en el hecho de que el modelo ha sido totalmente validado frente a estos datos en la literatura [Abdel-Hamid 1991].

Parámetro	Valor
Gestión de Recursos Humanos	
Dedicación de los técnicos al proyecto. Suponemos que el equipo de técnicos se encuentra empleado en dos proyectos de la organización por lo que la dedicación a este proyecto es del 50%	0.5
Retraso en la contratación. Se considera que desde que la dirección decide añadir nuevos técnicos al proyecto hasta que éstos comienzan a trabajar en el mismo transcurren 30 días	30 días
Tiempo medio de empleo de los técnicos en la organización. Se supone que los técnicos están en la organización durante un promedio de tres años.	1000 días
Dedicación media diaria de los técnicos con experiencia a la formación de los técnicos recién contratados. Suponemos que cada técnico experto dedica un 25% de su esfuerzo a esta actividad.	0,25
Tiempo medio de adecuación de los técnicos al proyecto. Se considera que tras 20 días, los técnicos recién contratados ya han adquirido la experiencia suficiente en el proyecto	20 días
Entorno de Desarrollo	
Número de líneas de código por cada tarea del proyecto	40 dsi
Porcentaje del esfuerzo total dedicado a tareas de desarrollo (producción y aseguramiento de calidad). Suponemos un 85%.	0.85
Entorno de Planificación	
Extensión máxima de la fecha de terminación. Suponemos que el proyecto no tiene plazo fijo y permitimos que se pueda modificar la fecha de terminación en un máximo del 80% de la fecha inicialmente planificada.	1.8

Estimaciones iniciales del proyecto	
Porcentaje de tareas infraestimadas inicialmente. Suponemos que hemos infraestimado este valor en un 35%	0.35
Esfuerzo estimado inicialmente para desarrollar el proyecto.	1111 t-d
Tiempo de finalización estimado inicialmente.	320 días
Porcentaje de infraestimación inicial en el número de técnicos. Suponemos que en el proyecto hemos infraestimado en un 40% este valor.	0,4

Tabla 4: Parámetros iniciales del modelo dinámico

El MDB lleva incorporado un sistema de métricas que le permite obtener información acerca del estado actual del proyecto, del esfuerzo invertido en las actividades de calidad y de los resultados que se han obtenido al aplicar dicho esfuerzo. Estas métricas se traducen en un conjunto de variables *cuantificables* que miden los costes de la calidad del software y cuyos valores se obtienen a través de la simulación del conjunto de ecuaciones diferenciales que integran el MDB. En concreto, las métricas que se utilizaron para evaluar los distintos costes de CoSQ son:

1. Costes por pérdida de calidad:
   · Costes de errores internos (CEI): Esfuerzo empleado en corrección.
   · Costes de errores externos (CEE): Esfuerzo necesario para corregir errores una vez entregado el producto. Este valor se calcula utilizando el número de errores en el producto al finalizar la etapa de pruebas y estimando que el coste de corrección de un error externo es el doble que el coste de corrección de un error interno (según [Boehm 1981]). Sin embargo, no nos ha sido posible evaluar esta métrica con la simulación del modelo dinámico ya, como hemos dicho anteriormente, no contempla la fase de mantenimiento y es imposible obtener esta información.

2. Costes para alcanzar la calidad:
   · Costes de valoración (CV): Esfuerzo empleado en pruebas.
   · Costes de prevención (CP): Esfuerzo empleado en revisiones/inspecciones de QA.

Con estas métricas, definimos el coste de la calidad del software como:

$$CoSQ = CP + CV + CEI + CEE$$

Los resultados de los costes obtenidos (en técnicos-día) para un proyecto de tamaño medio en condiciones normales se muestran en la Tabla 5 y gráficamente en la Figura 4.

CP (t-d)	*CEI (t-d)*	*CV (t-d)*	*CoSQ (t-d)*	*Coste Total (t-d)*
480,26	266,50	258,09	1004,85	1.950

Tabla 5: Costes y errores en un proyecto de tamaño medio

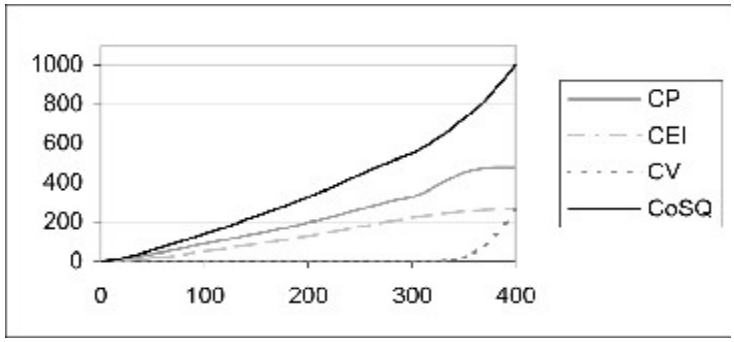


Figura 4: Costes de la calidad del software en un proyecto de tamaño medio

4 Impacto de diferentes políticas de gestión sobre CoSQ

A continuación, nos proponemos utilizar la simulación de este proyecto en condiciones distintas a ésta y observar cuál es la variación que sufre el coste de la calidad en relación con el coste total del proyecto. El estudio que hemos realizado es el siguiente:

- Determinar la influencia de la política de restricción en el plazo sobre el coste de la calidad.
- Averiguar la influencia de la combinación de políticas de restricción de plazo con las políticas de asignación de esfuerzo a las actividades de desarrollo sobre los costes de calidad.

Seguidamente exponemos los resultados obtenidos:

a) Política relativa la restricción del plazo. Simulamos el modelo dos veces: la primera para reproducir la situación de trabajar sin una presión de plazo importante y la segunda para contemplar el escenario con el mismo proyecto pero con una fuerte restricción en cuanto a la ampliación del plazo de entrega. La simulación de cada uno de estos escenarios se consigue modificando el valor de la variable que mide el porcentaje en que un PDS puede aumentar su fecha de terminación planificada inicialmente. Cuanto menor sea este porcentaje menor será también la posibilidad de aumentar la fecha de terminación del proyecto. Los resultados obtenidos se muestran en la Tabla 6.

	Plazo Fijo	Sin Plazo Fijo
Número de errores por KLDC	6,54	6,45
Coste Total (técnicos-día)	3.234,00	1.950,00
CP (técnicos-día)	678,86	480,26
CEI (técnicos-día)	314,24	266,50
CV (técnicos-día)	861,12	258,09
CoSQ (técnicos-día)	1.854,22	1.004,85

Tabla 6: Efectos de las políticas de restricciones de plazo sobre el CoSQ

En el caso de que el proyecto tenga restricciones en el plazo, el coste de la calidad representa un 57,3% del coste total del proyecto frente al 51,5% del caso en que el proyecto no posee restricciones en el plazo de entrega. Curiosamente, el número de errores cometidos por KLDC es muy similar en ambos escenarios, pero descubrimos que esto es así debido a la gran inversión realizada en actividades de detección (CP) y pruebas (CV) frente al caso en el que el proyecto no posee restricciones en el plazo de entrega.

b) En este apartado, simulamos el modelo para averiguar la influencia de la combinación de políticas de restricción de plazo con las políticas de asignación de esfuerzo a las actividades de desarrollo sobre los costes de la calidad. La asignación de esfuerzo al proyecto se realiza de la siguiente manera. Al esfuerzo total estimado para realizar el PDS, se le resta el que se dedica a la formación de los técnicos recién contratados. El esfuerzo resultante se reparte entre las actividades del proyecto de la siguiente forma: un porcentaje se dedica a las actividades de producción del software (diseño, codificación y aseguramiento de calidad) y el resto se asigna a las actividades de prueba. El esfuerzo necesario para aseguramiento de calidad se obtiene como un porcentaje del esfuerzo asignado a producción. Este porcentaje puede ser fijo e invariable a lo largo del ciclo de vida o puede ser variable según el estado del proyecto (por ejemplo, en situaciones de presión de plazo es común, aunque no aconsejable, disminuir el esfuerzo aplicado a calidad e invertirlo en actividades de producción).
Según lo visto, en este apartado se realizaron las simulaciones necesarias para evaluar los resultados de aplicar diferentes políticas de asignación de esfuerzo a las actividades de producción de software, en proyectos con y sin restricciones en los plazos de entrega. En todos estos casos, la asignación de esfuerzo para las actividades de aseguramiento de calidad se realizó atendiendo al estado actual del proyecto y a los factores de plazo, es decir, con un porcentaje variable. Los resultados obtenidos se muestran en la Tabla 7.

A la vista de los resultados mostrados podemos concluir dos factores. En primer lugar, la política de restricciones en los plazos de entrega es mucho más potente que la política de asignación de recursos a producción desde el punto de vista de coste de calidad. Ya sabemos que la política de restricción el plazo incrementa los costes del proyecto [Ramos 1997] porque la dirección generalmente opta por adquirir un mayor número de técnicos que permita acabar el proyecto en el plazo establecido. Este mayor número de técnicos produce un aumento del coste total del proyecto, y, como la asignación de esfuerzo para actividades de aseguramiento de calidad y pruebas no es más que un porcentaje del esfuerzo total del proyecto, los costes de estas actividades se ven igualmente aumentados.

Restricciones Plazo	*% de Esfuerzo Asignado a Producción*	*CoSQ*	*Nº Errores por KLDC*	*Coste Total*	*% CoSQ sobre Coste*
Sin Restricciones de Plazo	**Asignación Baja**	972,29	5,33	2.098	46,3%
	Asignación Alta	970,49	5,45	1.903	51%
Con Restricciones de Plazo	**Asignación Baja**	1.012,36	5,34	2.189	46,2%
	Asignación Alta	1.703	5,53	3.020	56,4%

Tabla 7: Resultados obtenidos combinando políticas de restricción de plazo y de asignación de esfuerzo a actividades de producción

Sin embargo, resulta curioso observar que frente a este notable incremento en el CoSQ de los proyectos con restricciones de plazo, no resulta igual de notable el efecto que tiene sobre la calidad del producto final. En efecto, si observamos la columna relativa al número de errores por cada mil líneas de código vemos que no se producen significativas diferencias en los proyectos con mayor CoSQ. Podemos encontrar una justificación atendiendo al hecho de que en proyectos con presión de plazo, el personal técnico comete muchos más errores debidos, por un lado, a la propia presión, y, por otro lado, al cansancio acumulado. Este aumento de los errores cometidos, requiere una mayor cantidad de actividades de detección y corrección, que no logran mejorar la calidad si comparamos estos proyectos con los que no tienen una importante presión del plazo, sino que únicamente contribuyen a garantizar un nivel de calidad que podríamos denominar como normal.

En cuanto a la dedicación del personal técnico a las actividades de producción de software los resultados, no por esperados, son menos interesantes. En los casos en los que el personal técnico tiene una dedicación baja en las actividades de producción, el CoSQ se sitúa en torno al 46% del coste total del proyecto, porcentaje sensiblemente menor que los proyectos con una asignación de esfuerzo alta. De entre los tres componentes del CoSQ evaluados, los que más contribuyen al valor final son los costes de valoración, ya que en proyectos con esta política de distribución de esfuerzo, son las actividades de pruebas las que resultan más beneficiadas. Sin embargo, si nos centramos en los valores nominales, podemos descubrir que una política de asignación de esfuerzo baja en las actividades de producción, independientemente de la política de plazo seguida, siempre incrementa el coste final del proyecto (ya que se necesita un mayor esfuerzo para terminarlo) y también el coste de las actividades de calidad del mismo, que no por ello consiguen mejorar el nivel de calidad final del proyecto.

5 Discusión y Conclusiones

CoQ es una técnica ampliamente utilizada en las industrias de manufacturas y de servicios para comunicar las iniciativas de mejora de calidad y para indicar los factores candidatos a una mejora de calidad. La técnica de CoSQ ofrece las mismas posibilidades para mejorar los PDS pero, sin embargo, no está totalmente adoptada por la comunidad de desarrollo de software. Las primeras aplicaciones de CoSQ mostraron que es posible reducir incluso los costes totales de un proyecto si planificamos y estimamos correctamente la inversión dedicada a aseguramiento de calidad.

Por otro lado, la simulación de modelos dinámicos para PDS había sido utilizada para obtener la relación entre los errores cometidos en un proyecto y el gasto empleado en aseguramiento de la calidad [Abdel-Hamid 1991]. En estos estudios iniciales sólo se extraen como conclusiones que la política de asignación de recursos para actividades de aseguramiento de calidad influye de manera clara sobre los costes del proyecto y que este efecto se puede recoger a través de la simulación de un modelo dinámico. Sin embargo, no se conoce en la literatura ningún estudio que utilice, de forma conjunta, la simulación de modelos dinámicos para PDS y las métricas de CoSQ para predecir el coste de la calidad del software de la forma en que aquí se ha realizado.

A nuestro juicio, la unión de estas dos técnicas ofrece tres importantes y ventajosas posibilidades:

1ª) Proporciona un mecanismo para evaluar la evolución de los proyectos a partir de una inversión en calidad de software determinada, posibilitando la asignación correcta de recursos y la experimentación sin coste en algo tan complejo como la gestión de proyectos.

2ª) Ofrece una medida objetiva para la comparación de escenarios distintos para un mismo proyecto,

para proyectos de una misma organización e incluso para proyectos de distintas organizaciones de desarrollo.

3ª) La utilización conjunta de CoSQ y la simulación de proyectos permite la obtención de los valores de las métricas durante la ejecución del proyecto, es decir, no es necesario esperar al final del proyecto para obtener la evolución del CoSQ a lo largo de todo el ciclo de vida, sino, lo que es más importante, es totalmente posible simular alguna de las fases del ciclo de vida, por ejemplo la fase de diseño, con unos determinados valores y observar cuál es la evolución del producto y de su calidad y en ese mismo momento ensayar posibles políticas de gestión alternativas que optimicen los resultados finales.

En definitiva, la combinación de las técnicas descritas anteriormente nos permitirá ensayar diferentes alternativas de gestión mediante el análisis de los resultados obtenidos. Ello posibilita el establecimiento de un debate con otros técnicos en gestión de PDS o bien entrenar al personal novel en la materia.

No obstante, consideramos que es altamente recomendable el desarrollo de un modelo dinámico que nos permita reproducir más allá de la fase de pruebas, que incluya a las actividades de mantenimiento, con el fin de poder evaluar la cuarta métrica propuesta de CoSQ, la cual consideramos de vital importancia debido a la crucial información que contiene y que no es otra que la calidad encontrada en el producto por el cliente. Esta medida nos proporcionaría el coste de detectar y corregir los errores del producto en su ambiente de explotación.

Referencias

[Abdel-Hamid 1991] Abdel-Hamid, T.; Madnick E., *Software Project Dynamics: An Integrated Approach*, Prentice-Hall, 1991.

[Alberts 1976] Alberts, D.S., "The Economics of Software Quality Assurance", *National Computer Conference*, 1976, pp. 433-441.

[Boehm 1981] Boehm, B.W., *Software Engineering Economics*, Prentice-Hall, 1981.

[Crosby 1979] Crosby, P.B., *Quality is Free: The Art of Making Quality Certain*, McGraw-Hill, 1979.

[Dion 1993] Dion, R., "Process Improvement and the Corporate Balance Sheet", *IEEE Software*, Julio, 1993, pp. 28-35.

[Eberlein 1989] Eberlein, R.L., "Simplification and Understanding of Models." System Dynamics Review , 5:1 (1989), pp. 51-68.

[Hersch 1993] Hersch, A., "Where's the Return on Process Improvement", *IEEE Software*, Julio, 1993.

[Juran 1996] Juran, J.M.; A History of Managing for Quality. McGraw-Hill, 1996.

[Knox 1993] Knox, S.T., "Modeling the Cost of Software Quality", *Digital Technical Journal*, 5:4 (fall 1993), pp. 9-16.

[Plunkett 1990] Plunkett, J.J.; Dale, B.G., "Quality Costing", en *Managing Quality*, J.J. Plunkett y B.G. Dale, ed., Phillip Allan, 1990.

[Ramos 1997] Ramos, I.; Ruiz, M., "Estudio Dinámico de la Política de Plazo Fijo sobre Proyectos de Desarrollo de Software", *XIII Congreso Nacional de Ingeniería de Proyectos*, 1997, pp. 705-710.

[Ruiz 2000] Ruiz, M.; Ramos, I., "A Dynamic Estimation Model for the Early Stages of a Software Project", International Workshop on Software Process Simulation Modeling. ProSim 2000. Londres, julio de 2000.