CONTROL DE CALIDAD EN EL SOFTWARE

JOSE HERNANDO BAHAMON L.

Ingeniero en Electrónica y Telecomunicaciones de la Universidad del Cauca. Profesor Investigador en el área de microprocesadores en la facultad de Electrónica de la Universidad del Cauca. Gerente Nacional de Soporte a Clientes en la División de Computadores de Carvajal S.A. Participante en numerosos cursos de actualización y profundización en computadores, sistemas operativos, herramientas de programación de cuarta generación, ingeniería de software, etc. Jefe del Departamento de Sistemas del ICESI. Profesor universitario. Conferencista. Investigador y Asesor de Empresas en Sistemas y Computación.

El concepto de calidad total aplicado por los japoneses como estrategia de desarrollo a partir de la Segunda Guerra Mundial, con el fin de recuperar su economía y tener una presencia a nivel internacional, ha empezado a popularizarse a nivel mundial y es el tema obligado de las naciones, organizaciones, entidades e individuos que buscan consolidación y presencia en los mercados del mundo.

El concepto de calidad total propende por la búsqueda de la excelencia en todo lo que el hombre, la sociedad y las organizaciones realizan.

Este concepto puede entonces aplicarse al desarrollo de sistemas de información basados en equipos de procesamiento de datos y en programas diseñados por el hombre. Hoy en día las investigaciones en el área de la ingeniería de software se centran en el desarrollo de metodologías que garanticen y controlen la calidad en el software construido.

El presente documento busca ilustrar, además del concepto de calidad en el software, las actividades necesarias para controlar y garantizar la calidad de los sistemas de información que se implementen. El problema principal para garantizar la calidad en el software está en la concepción de la gran mayoría de las personas cuando suponen que la garantía de calidad es algo que se impone bajo una medida que se obtiene al finalizar un proyecto de software. El control de calidad en el software se funda-

menta en el principio de que la calidad se construye a través de un proceso continuo de desarrollo, verificación (revisión) y optimización en diferentes etapas.

El control de calidad en el software, denominado SQA ("Software Quality Assurance"), se basa en las siguientes actividades:

1) Uso de métodos y herramientas de análisis, diseño, codificación y prueba.
2) Revisiones técnicas formales, que se aplican durante cada paso de la Ingeniería de software.
3) Estrategia de prueba multiescalada.
4) Control de la documentación del software y de los cambios realizados.
5) Procedimientos que aseguren un ajuste a los estándares de desarrollo.
6) Mecanismos de medida de la calidad ("métricas").

DEFINICION DE CALIDAD EN EL SOFTWARE

Se han formulado muchas definiciones sobre el concepto de calidad en el software. Para no transcribir estas definiciones en el presente documento tratemos de responder la pregunta ¿qué es calidad en el software? Seguramente la primera respuesta en que pensaría la mayoría de las personas es:

La calidad en el software está en relación directa con el cumplimiento de los requerimientos formulados por el usuario, de tal forma que si un programa no cumple con alguno de estos requerimientos es un software de baja calidad.

Aunque el criterio de cumplimiento de los requerimientos es un factor importante, no es el único factor, ya que existen condiciones implícitas que el software debe cumplir como son eficiencia, seguridad, integridad, consistencia, etc.; por lo tanto no podemos afirmar que un software es de alta calidad cuando cumple con los requerimientos del usuario, pero:

- No es eficiente al utilizar los recursos de la máquina (programas muy lentos).
- O no es confiable; los resultados que entrega varían, no son siempre iguales al procesar los mismos datos.
- O no es fácil de utilizar.
- O no es seguro.
- O no es fácil hacerle mantenimiento.

La calidad en el software es una mezcla compleja de ciertos factores que varían de acuerdo con el usuario y con los tipos de aplicación.

Podemos resumir el concepto de la calidad en el software en los siguientes puntos:

1) Los requerimientos del usuario sobre un programa son los fundamentos desde los que se mide la calidad. La falta de concordancia con estos requerimientos es una falta de calidad.
2) Los estándares especificados definen un conjunto de criterios de desarrollo que guían la forma como se aplica la ingeniería de software; si no se siguen estos estándares, probablemente se obtendrá software de baja calidad.
3) Existe un conjunto de requerimientos implícitos que a menudo no se mencionan (eficiencia, facilidad de uso, facilidad de mantenimiento). Si el software falla en alcanzar los requerimientos implícitos, la calidad en el software queda en entredicho.

FACTORES Y CRITERIOS QUE DETERMINAN LA CALIDAD EN EL SOFTWARE

Los elementos básicos empleados para medir la calidad en el software se denominan factores; éstos pueden clasificarse en dos grandes categorías:

a) Factores que pueden ser medidos directamente:
(Nº de errores/unidad tiempo).

b) Factores que sólo pueden ser medidos indirectamente; valores subjetivos (Ejemplo: facilidad de uso).

Según los estudios realizados por J.A. McCall y P.K. Richards para la RADC ("Rome Air Development Center"), los factores de calidad se pueden agrupar de acuerdo con tres aspectos importantes de todo programa, como son sus características operacionales, su capacidad de soportar los cambios y su adaptabilidad a nuevos entornos.

La clasificación sugerida por J.A. McCall en su libro *Factors in Software Quality,* se ilustra en la tabla Nº 1, y la descripción de cada factor se ilustra en la tabla Nº 2.

En la mayoría de los casos los factores son difíciles de medir, para facilitar el proceso de cuantificar la calidad, McCall propone dividir los factores en sus características independientes o criterios medibles. Las razones fundamentales para dividir los factores son:

1) Los criterios ofrecen una definición más concreta y completa de los factores.

2) Los criterios comunes a dos o más factores ayudan a ilustrar la interrelación entre los factores.

3) Los criterios son medibles y verificables a través de métricas (valor numérico de la medida de calidad).

La lista de criterios se ilustra en la gráfica 1, y la relación entre los criterios y los factores se muestra en la gráfica 2.

NEGOCIACIONES ENTRE LOS FACTORES DE CALIDAD

Si observamos los factores de calidad podemos ver que el incrementar un factor puede causar efectos negativos (decremento) en otros factores. Por ejemplo, si nosotros solicitamos que el factor de facilidad de uso sea muy alto seguramente esto se logrará a expensas de disminuir la eficiencia del programa.

Las relaciones negativas entre factores se ilustran en la gráfica 3.

Es necesario definir, basados en la naturaleza y tipo de software a producir, los factores que el usuario considere de mayor importancia y estimar el impacto negativo que se pueda causar en otros factores, con el fin de establecer una negociación hasta obtener la ponderación deseada en cada factor. Esta actividad de negociación debe establecerse en la etapa de formulación de los requerimientos.

TABLA 1

Aspecto	**Factor**
Operación del producto	Cumplimiento Exactitud Eficiencia Integridad Facilidad de uso
Revisión del producto	Facilidad de mantenimiento Facilidad de prueba Flexibilidad
Transición del producto	Portabilidad Reusabilidad Interoperatividad

METRICAS DE CONTROL DE LA CALIDAD EN EL SOFTWARE

Se define como métrica el valor asociado con la respuesta a una pregunta formulada en una revisión para evaluar o establecer un atributo o un requerimiento de un criterio o subcriterio relacionado con un factor. Por ejemplo:

Un criterio del factor de calidad "Eficiencia" es "Ejecución eficiente" y un atributo de este subcriterio sería "datos agrupados para procesamiento eficiente". En una revisión para evaluar este subcriterio se podría formular la siguiente pregunta: "¿Están los datos agrupados para permitir un procesamiento eficiente"?

Si la respuesta a la pregunta es "sí", podemos calificar con 1 en la hoja de chequeos este subcriterio; si la respuesta es "no" lo calificamos con 0.

El valor de la métrica para el factor de calidad que está siendo juzgado será la suma de todos los valores obtenidos por criterios/subcriterios divididos por el número de preguntas aplicadas.

En los estudios realizados por McCall se establece un conjunto de métricas para los diferentes criterios y subcriterios. En la gráfica 4 se ilustran algunas de estas métricas.

TABLA 2

Factor calidad	**Definición**
Cumplimiento	El grado en que un programa satisface sus especificaciones y consigue los objetivos de la misión encomendada por el cliente.
Fiabilidad	El grado en que se puede esperar que un programa lleve a cabo sus funciones esperadas con la precisión requerida.
Eficiencia	La cantidad de recursos de hardware y de código requerido por un programa para realizar su función.
Integridad	El grado en que puede controlarse el acceso al software o a los datos por personas no autorizadas.
Facilidad de uso	El esfuerzo requerido para aprender, trabajar, preparar la entrada e interpretar la salida de un programa.
Facilidad de mantenimiento	El esfuerzo requerido para localizar y arreglar un error en un programa.
Facilidad de prueba	El esfuerzo requerido para probar un programa de forma que se asegure que realiza la función requerida.
Portabilidad	El esfuerzo requerido para transferir el programa desde una configuración de hardware o sistema operativo a otro.
Reusabilidad	El grado en que un programa (o partes de él) se pueden reutilizar en otras aplicaciones.
Facilidad de interoperación	El esfuerzo requerido para acoplar un sistema a otro.

GRAFICA 1

Facilidad de auditoría	Facilidad con que se puede comprobar la conformidad con los estándares.
Exactitud	La precisión en los cálculos y el control.
Normalización de las comunicaciones.	El grado en que se usan el ancho de banda, los protocolos y las interfases estándar.
Completitud	El grado en que se ha conseguido la total implementación de las funciones requeridas.
Concisión	Lo compacto que es el programa en términos de líneas de código.
Consistencia	El uso de un diseño uniforme y de técnicas de documentación.
Estandarización datos	El uso de estructuras de datos y de tipos datos estándar.
Tolerancia de errores	El daño que se produce cuando el programa encuentra un error.
Eficiencia en la ejecución	El rendimiento en tiempo de ejecución de un programa.
Facilidad de expansión	El grado en que se puede ampliar el diseño arquitectónico de datos.
Generalidad	La amplitud de aplicación potencial de los componentes del programa.
Independencia del hardware	El grado en que el software es independiente del hardware que usa.
Instrumentación	El grado en que el programa muestra su propio funcionamiento e identifica errores que aparecen.
Modularidad	Independencia funcional de los componentes del programa.
Facilidad de operación	Grado de facilidad de operación.
Seguridad	La disponibilidad de mecanismos que controlen o protejan los programas o los datos.
Autodocumentación	El grado en que el código fuente proporciona documentación significativa.
Simplicidad	El grado en que un programa puede ser entendido sin dificultad.
Facilidad de trazo	La posibilidad de seguir la pista de la representación del diseño de los componentes reales del programa hacia atrás.
Formación	El grado en que el software ayuda a permitir que nuevos usuarios apliquen el sistema.

GRAFICA 2

Métrica de calidad del software / Factor de calidad	Corrección	Fiabilidad	Eficacia	Integridad	Facilidad de mantenimiento	Flexibilidad	Facilidad de prueba	Portabilidad	Reusabilidad	Interoperabilidad	Facilidad de uso	
Facilidad de auditoría				X			X					
Exactitud		X										
Normalización de las comunicaciones										X		
Completitud	X											
Complejidad		X				X	X					
Concisión			X		X	X						
Consistencia	X	X			X	X						
Estandarización en los datos										X		
Tolerancia de errores		X										
Eficiencia en la ejecución			X									
Facilidad de expansión						X						
Generalidad						X		X	X	X		
Indep. del hardware								X	X			
Instrumentación				X	X		X					
Modularidad		X			X	X	X	X	X	X		
Facilidad de operación			X								X	
Seguridad				x								
Auto-documentación					X	X	X	X	X			
Simplicidad		X			X	X	X					
Indep. del sistema								X	X			
Facilidad de traza	X											
Formación											X	

Gráfica 3

	Cumplimiento	Fiabilidad	Eficiencia	Integridad	Fac. de uso	Fac. de mantenimiento	Fac. de prueba	Flexibilidad	Portabilidad	Reusabilidad
Cumplimiento										
Fiabilidad	○									
Eficiencia			●							
Integridad			●							
Fácil. de uso	○	○	●	○						
Fácil. de mantenimiento	○	○	●		○					
Fácil. de prueba	○	○	●		○	○				
Flexibilidad	○	○	●	●	○	○	○			
Portabilidad			●			○	○			
Rehusabilidad		●	●	●		○	○	○	○	
Facilidad de Interop.			●	●					○	

● = Impacto negativo
○ = Impacto positivo
Blanco = no hay relación

MEDIDA DE CONCISION Y COMPLEJIDAD DE HALSTEAD

La teoría de complejidad del software propuesta por Maurice Halstead en su libro *Elements of Software Science*, es probablemente la más conocida y estudiada. Esta teoría está basada en la suposición fundamental de que la medida de un programa bien estructurado está en función de sus operandos y operadores.

Estudios realizados en las universidades sobre un gran número de programas existentes han confirmado la validez de las premisas de Halstead.

La medida de concisión propuesta por Halstead involucra el cálculo de dos variables:

A) La longitud calculada (Nc):

$$Nc = n_1 \log_2 n_1 + n_2 \log_2 n_2$$

B) La longitud observada (No):

$$No = n1 + n2$$

donde

n1 = Nº total de operadores distintos en el programa
n2 = Nº total de operandos distintos en el programa

La medida de concisión (MC) se obtiene por medio de la siguiente fórmula:

$$MC = 1 - \frac{Nc - No}{No}$$

Otra medida propuesta por Halstead es el cálculo del "Esfuerzo", entendido como la cantidad de tiempo requerido para escribir, modificar, mantener o depurar un código; para obtener esta medida se deben evaluar las siguientes variables:

A) El volumen (V): número de bits requeridos al especificar el programa.

$$V = (N1 + N2) \log_2 (n1 + n2)$$

B) El volumen potencial (V*): número de bits para la forma más compacta del programa.

GRAFICA 4

Criterio completitud

atributos:
1) Referencia sin ambigüedad (entrada, salida, función)
2) Definidas todas las referencias externas, calculadas u obtenidas por fuentes externas.
3) Definidas todas las funciones usadas
4) Definidas todas las funciones referenciadas.
5) Definidas todas las condiciones para cada punto de decisión.
6) Definidos todos los parámetros para la secuencia de llamadas a procesos.
7) Todos los problemas reportados están resueltos.
8) Diseño está de acuerdo con los requerimientos.
9) La codificación está de acuerdo con el diseño

V. métrica criterio: $\dfrac{\sum_{1}^{9} \text{valor c/atributo}}{9}$

- Criterio: Consistencia

Subcriterio : Procedimiento consistente

atributos :
a) Representación estándar en el diseño*
b) Secuencia de llamada a módulo según lo establecido*
c) Entrada / salida según lo establecido*
d) Manejo de errores según el estándar*

Valor métrica:

Valor métrica: $1 - \dfrac{\text{Número módulos que violan la regla}}{\text{Número total de módulos}}$

(*) Significa que la fórmula de evaluación se aplica a cada atributo.

$$V^* = (n1 + n2^*) \log_2 (n1 + n2^*)$$

donde

N1 = Nº total de ocurrencias de los operadores

N2 = Nº total de ocurrencias de los operandos

n2* = Nº de parámetros de I/O en el procedimiento.

Usando el volumen y el volumen potencial se puede calcular el esfuerzo como:

$$\text{esfuerzo} = \frac{\text{Volumen}^2}{\text{Volumen potencial}} = \frac{V^2}{V^*}$$

Estas métricas propuestas por Halstead permiten al grupo de control de garantía del software obtener valores no subjetivos de la calidad de un programa.

Existen otras métricas como la medida de complejidad ciclomática propuesta por Tom McCabe en su artículo *A "Software Complexity Measure"*, publicado en la revista *IEEE Trans. Software Engineering,* vol. 2, diciembre de 1976; pero el objetivo de este documento no es ilustrar en detalle todas las métricas propuestas.

ACTIVIDADES DEL CONTROL DE CALIDAD DEL SOFTWARE

Hasta el momento hemos mencionado el concepto de calidad en el software y algunas técnicas empleadas para establecer una medida cuantitativa de la calidad.

La historia de la garantía de la calidad en el desarrollo de programas y sistemas arranca en los años 50 y 60, en donde la calidad era responsabilidad únicamente del programador. Durante los años 70 se introdujeron estándares de garantía de calidad para el software en los contratos militares y se incorporaron las metodologías para el desarrollo de sistemas.

Actualmente la responsabilidad de la garantía de calidad del software no es función de una persona; en esto están comprometidos los ingenieros de análisis y diseño, los gestores y coordinadores del proyecto, los usuarios, los programadores y todas las personas involucradas en el desarrollo del proyecto.

La garantía de calidad en el software no es una certificación impuesta luego de haber desarrollado un programa. Es un proceso que involucra las siguientes actividades:

1) Aplicación de metodologías de ingeniería de software para conseguir una especificación y un diseño de alta calidad.
2) Realización de revisiones técnicas formales.
3) Prueba del software.
4) Ajuste a los estándares de la organización.
5) Control de cambios y modificaciones (mantenimiento).
6) Mediciones.
7) Registro e informes.

La garantía de calidad en el software comienza realmente con la aplicación de una metodología formal para enfrentar las etapas de análisis y diseño del sistema a construir; luego de creada la especificación del sistema (o prototipo), se debe garantizar su calidad.

La actividad que nos permite garantizar la calidad es la revisión técnica formal realizada por el grupo de control de calidad.

Los objetivos de la revisión técnica formal son:

1) Descubrir errores en la función, la lógica o la implementación de cualquier representación del software.
2) Verificar que el software bajo revisión alcanza los requerimientos.
3) Garantizar que el software ha seguido los estándares predefinidos.
4) Conseguir un software que sea desarrollado en forma uniforme.
5) Propender por que los proyectos sean manejables.

La revisión técnica formal es un proceso que se aplica a cada una de las fases del desarrollo del sistema en el momento en que el grupo de trabajo considera terminada su labor en esa fase.

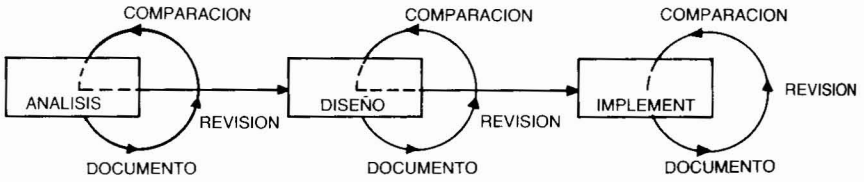
Como resultado de la revisión técnica formal se obtiene una autorización para que el grupo pueda continuar con la fase siguiente, o una recomendación de no continuar hasta realizar las modificaciones y ajustes al proceso en la fase bajo revisión (Gráfico 5).

Una vez que se ha terminado la implementación, se inicia la fase de pruebas del software. Durante esta fase se diseñan casos de prueba que ayudan a la detección de errores producidos en las fases anteriores y no detectados durante la revisión técnica formal.

Para muchos grupos de desarrollo las pruebas del software son consideradas una "red de seguridad" para la garantía de la calidad.

Una de las principales amenazas para mantener la calidad de un software, es el proceso de mantenimiento a través

GRAFICA 5



Sistema de control de la garantía de software

del cual se originan cambios que pueden introducir errores o crear efectos laterales que propaguen errores.

El proceso de control de cambios contribuye directamente a mantener la calidad de un programa al formalizar las peticiones de mantenimiento, evaluar la naturaleza del cambio y controlar el impacto de éste en el resto del programa.

Finalmente la medición de la calidad se fundamenta en las métricas, las cuales nos permiten cuantificar y tener valores comparativos sobre el comportamiento y la eficiencia, en el desarrollo de programas y sistemas para la organización.

CONCLUSION:

En los últimos años se están adelantando esfuerzos por parte de los grupos de investigación en el área de ingeniería de software con el fin de formular estrategias, procedimientos de control y medida de la calidad en el software.

Desafortunadamente algunos de estos procedimientos son bastante complejos de implementar; por esta razón la mayoría de las organizaciones en sus departamentos de sistemas no están utilizando el enfoque de control de calidad en el software.

BIBLIOGRAFIA

David N. Card, *Software product assurance: measurement and control.* Inf & Software Technol. (Agosto 1988).

B. A Kitchenham and S.J. Linkman, *Design metrics in pratice,* Inf & Software Technol. (Mayo 1990).

D. Ince, *Software metrics: introduction,* Inf & Software Technol. (Mayo 1990).

David N Card, *Software quality engineering,* Inf & Software Technol. (Febrero 1990).

James Vincent, Albert Waters, John Sinclair, *Software Quality Assurance,* volumen 1, Prentice Hall (1988).