Calidad del software (I)

Los primeros años de la era informática se vieron marcados por el desafío del desarrollo del hardware de los ordenadores. Este desafío se ha visto superado por los vertiginosos avances de la microelectrónica. Hoy en día el principal problema al que nos enfrentamos es la llamada "crisis del software". La causa principal de esta crisis es el aumento de la complejidad de las aplicaciones sin la adopción de los procesos adecuados de desarrollo que contribuirían a obtener un producto de mayor calidad y menor coste. En este artículo se van a presentar cuáles son esas claves para conseguir un software de mayor calidad.

Yolanda González Arechavala
Licenciada en Informática por la Universidad del País Vasco. Es investigadora en Formación del Instituto de Investigación Tecnològica y profesora en el Departamento de Sistemas Informáticos, ambos de la ETS de Ingeniería de la UP-Co. Este trabajo ha sido parcialmente financiado con una beca de la Asociación de Ingenieros del ICAI.

Fernando de Cuadra García
Dr. Ingeniero Industrial del ICAI. Es Profesor Propio Agregado de la UPCo, Director de la Escuela Técnica Superior de Ingeniería (ICAI) e Investigador del Instituto de Investigación.

Introducción

La aparición en la prensa de frases como "El desarrollo del software se ha convertido en un tema crítico para las modernas organizaciones, y por tanto para toda la sociedad" (Fitzgerald & O'Kane, 1999) o "Que el desarrollo del software sea un éxito implica mucho más que escribir código. La mayoría de las organizaciones de desarrollo del software comprenden ahora que el éxito de los procesos depende de la organización y estructura de la empresa, el control de proyectos y los procedimientos y estándares software" (Dutta, Lee, & Van Wassenhove, 1999) ponen de relieve la importancia que actualmente se le da al desarrollo del software.

Con el fin de solucionar este problema (que se ha venido llamando la crisis del software) surge la disciplina llamada ingeniería del software que se define como (Pressman, 1995):

"Establecimiento y uso de principios de ingeniería robustos, orientados a obtener software económico que sea fiable y funcione de manera eficiente sobre máquinas reales".

Una de la áreas de trabajo de la ingeniería del software que más auge está teniendo actualmente es la calidad, aunque realmente sea una disciplina poco extendida. Seleccionando una de las posibles definiciones de (Nistal Rosique, 1999):

TABLA 1. FCTORES DE CALIDAD DE MCCALL

Aspecto que trata	Factor Calidad	Relacionado con ...
Operación del Producto	Corrección	Grado en que un programconsigue los objetivos de la misión encomendada por el cliente
	Fiabilidad	Probabilidad de operación libre de fallos de un programa de computadora en un entorno determinado durante un tiempo específico
	Eficiencia	Cantidad de recursos de computadora y de código requeridos por un programa para llevar a cabo sus funciones
	Integridad	Grado en que puede controlarse el acceso al software o a los datos, por personal no autorizado
	Facilidad de uso	Esfuerzo requerido para aprender un programa, trabajar con él, preparar su entrada e interpretar su salida
Revisión del producto	Facilidad de mantenimiento	Esfuerzo requerido para localizar y arreglar un error en un programa
	Flexibilidad	Esfuerzo requerido para modificar un programa operativo
	Facilidad de prueba	Esfuerzo requerido para probar un programa de forma que se asegure que realiza la función requerida
Transición del producto	Portabilidad	Esfuerzo requerido para transferir el programa desde un hardware y/o entorno de sistemas de software a otro
	Reusabilidad	Grado en que un programa (o partes de un programa) puede reusarse en otras aplicaciones
	Facilidad de interoperación	Esfuerzo requerido para acoplar un sistema a otro

"Calidad del Software consiste en desarrollar productos lógicos que, cumpliendo las normas, satisfagan las necesidades del usuario, los requisitos implícitos (a menudo, no mencionados) y que tiendan a cero defectos."

Para conseguir un software de calidad es necesario realizar una serie de tareas a lo largo de todo el proceso de desarrollo de la aplicación. Es lo que se conoce como la garantía de calidad del software (SQA: "Software Quality Assurance"). Según Pressman:

"La garantía de calidad del software es un diseño planificado y sistemático de acciones que se requieren para asegurar la calidad del software"

En los siguientes apartados se van estudiar en primer lugar los factores que afectan a la calidad del software. Posteriormente, se estudiarán las tareas relacionadas con la garantía de calidad del software. Con esto se contemplará la primera parte de este artículo. En la segunda parte se analizarán dos de estas tareas, como son las métricas empleadas so-

TABLA 2. MÉTRICAS PROPUESTAS POR MCCALL

Métrica	Relacionado con ...
Facilidad de auditoría	La facilidad con que se puede comprobar la conformidad con los estándares
Exactitud	La precisión de los cálculos y del control
Normalización de las comunicaciones	El grado en que se usa el ancho de banda, los protocolos y las interfaces estándar
Completitud	El grado en que se ha conseguido la total implantación 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 a lo largo de todo el programa
Estandarización en los datos	El uso de estructuras de datos y de tipos estándar a lo largo de todo el programa
Tolerancia de errores	El daño que se produce cuando el programa detecta una situación erronea
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 o procedimental
Generalidad	La amplitud de aplicación potencial de los componentes del programa
Independencia del hardware	El grado en que el software es independiente del hardware sobre el que opera
Instrumentación	El grado en que el programa muestra su propio funcionamiento e identifica errores que aparecen
Modularidad	La independencia funcional de los componentes del programa
Facilidad de operación	La facilidad de utilización de un programa
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
Independencia del sistema de software	El grado en que el programa es independiente de características no estándar del lenguaje de programación, de las características del sistema operativo y de otras restricciones del entorno
Facilidad de traza	La posibilidad de seguir la pista a la representación del diseño o de los componentes reales del programa hacia atrás, hacia los requisitos
Formación	El grado en que el software ayuda a permitir que nuevos usuarios empleen el sistema

bre calidad del software y los estándares que existen en la actualidad. Como punto final, se muestran las restricciones que tienen algunas de las técnicas comentadas en el artículo para garantizar la calidad del software.

Factores de Calidad

Según (Pressman, 1995), basándose en los estudios de (McCall, 1977), los factores que afectan a la calidad del software se centran en tres aspectos importantes de un producto software: sus características operativas, su capacidad de soportar los cambios (revisión del producto) y su adaptabilidad a nuevos entornos (o transición del producto). En la Tabla I se muestran los factores de calidad definidos por McCall. Estos factores de calidad de McCall, a pesar de haberse definido hace mucho tiempo, conservan en gran medida su vigencia (salvo para el caso de la programación orientada a objetos).

Es difícil (y en algunos casos imposible) desarrollar medidas directas de los anteriores factores de calidad. Por tanto, se define un conjunto de métricas usadas para evaluar los factores de calidad. Según de qué factor se trate, se utilizarán unas determinadas métricas ponderadas para determinar ese factor. Las métricas (algunas de ellas medidas subjetivas) que utiliza McCall para evaluar los distintos factores

de calidad se muestran en la Tabla 2.

Garantía de Calidad

La garantía de calidad del software comprende una gran variedad de tareas, asociadas a siete actividades principales:

Métodos y herramientas de análisis, diseño y codificación

La calidad del software debe estar diseñada para el producto o sistema, no es algo impuesto a posteriori. Por esta razón, la garantía de calidad del software comienza realmente con un conjunto de herramientas y métodos técnicos que ayudan al analista a conseguir una especificación y un diseño de alta calidad.

En un proceso de desarrollo software, existen una serie de fases que vienen determinadas por las tareas básicas que hay que realizar para obtener el producto software. Se definen distintos ciclos de vida de acuerdo a la evolución del software a lo largo de dichas fases. Las fases más típicas son:

Requisitos del Sistema: también llamada análisis de requisitos, especificación o diseño conceptual o diseño de alto nivel. Según (Durán Toro, Bernárdez Jiménez, Corchuelo Gil, & Toro Bonilla, 2000) en (IEEE, 1990) se define:

"Requisitos: (a) una condición o capacidad que un usuario necesita para resolver un problema o lograr un objetivo. (b) una condición o capacidad que debe tener un sistema o un componente de un sistema para satisfacer un contrato, una norma, especificación u otro documento formal. (c) una representación en forma de documento de una condición o capacidad como las expresadas en (a) o (b)."

La ingeniería de requisitos se define como:

"Todas las actividades relacionadas con: (a) identificación y documentación de las necesidades del cliente y usuarios, (b) creación de un documento que describe la conducta externa y las restricciones asociadas del sistema que satisfará dichas necesidades. (c) análisis y validación del documento de requisitos para asegurar su consistencia, viabilidad y que sea completo. (d) evolución de las necesidades."

Existen varias propuestas en cuanto a las actividades que conlleva la ingeniería de requisitos. Por su claridad, se ha utilizado la presentada en (Durán Toro et al., 2000), que comprende tres actividades principales: obtención, análisis y validación. La mayor parte de las normas y autores asumen que el proceso de obtención de los requisitos, su análisis y validación es iterativo por naturaleza, ya que se reconoce que es prácticamente imposible obtener todos los requisitos y que éstos sean correctos sin tener que volver atrás en algún momento del proceso. Las actividades son:

- Obtención ("Elicitation"[1]) de Requisitos: los clientes, compradores o usuarios del sistema a desarrollar descubren, revelan, articulan y entienden sus propios requisitos. Los requisitos se obtienen con entrevistas, cuestionarios, reuniones de las distintas partes implicadas y diversas técnicas cuyo resultado deben ser los requisitos orientados al cliente o también llamados requisitos - C, que serán analizados en la siguiente actividad.

- Análisis de requisitos: se debe razonar sobre los requisitos-C para comprender mejor el problema, detectar conflictos o inconsistencias, combinar requisitos relacionados e identificar nuevos requisitos, normalmente mediante la construcción de modelos, en la que podrían participar aquellos clientes y usuarios con los conoci-

[1] El término en inglés "elicitatión" se utiliza en la obtención de requisitos para definir la actividad realizada por el analista con el fin de proteger la información que de las necesidades del sistema tienen los usuarios finales, los expertos y los clientes. Además, esta actividad debe ser capaz de mostrar el conocimiento tácito, aquello que los usuarios no comentan por olvidarlo o considerarlo obvio.

mientos apropiados. El producto de esta actividad son los requisitos orientados al desarrollador o requisitos-D, y en algunas ocasiones, un prototipo.

En esta fase ya se están tomando decisiones de cómo hacer el sistema ya que se realiza una descomposición del sistema en subpartes siguiendo la técnica de "divide y vencerás".

- Validación de requisitos: los clientes y usuarios deben confirmar que los requisitos-C, una vez analizados, son válidos, correctos y completos, mediante las inspecciones de los documentos generados y mediante la evaluación del prototipo, proceso que por lo general conlleva la obtención de nuevos requisitos. Además será necesario validar que los requisitos -D encajan con los requisitos -C.

La nomenclatura de los requisitos (requisitos-C para los del cliente y requisitos-D para los del desarrollador) fue presentada en (Brackett, 1990) y está siendo aceptada por algunos autores como (Durán Toro et al., 2000). La manera habitual de expresar los requisitos-C es el lenguaje natural, que puede acompañarse del uso de plantillas y patrones lingüísticos para facilitar su uso. La forma de expresar los requisitos -D suele ser mediante un modelo construido con técnicas estructuradas (DFD, ERD, etc), técnicas orientadas a objetos (OMT, UML, etc) o técnicas formales..

Dado que diversos estudios han revelado que el alto índice de fracasos en los proyectos de desarrollo software tiene como principales causas actividades relacionadas con los requisitos (falta de participación de los usuarios, requisitos incompletos y frecuentes cambios de los requisitos iniciales), es lógico que muchos estudios se basen en esta etapa, por lo que las publicaciones relativas a la ingeniería de requisitos han sido abundantes en los últimos años (por ejemplo, los números especiales de las revistas "IEEE Software" Marzo/Abril 1998 o el de "Communications of the ACM" Diciembre 1998 así como diversos libros). Además, existen multitud de métodos de modelado en general que se utilizan para el modelado de los requisitos-D. Otro ejemplo es el proyecto MENHIR: Metodologías, Entornos y Nuevas Herramientas para la Ingeniería de Requisitos (Proyecto de la CICYT TIC 97-0593-C05-0).

Diseño: también llamado diseño arquitectural o diseño de bajo nivel.

Partiendo del modelo de análisis de requisitos obtenido en la fase anterior, se transforma éste en un conjunto de entes físicos (hardware) y entes lógicos (software) inter-relacionados entre sí que no tienen por qué conservar la misma estructura que en el modelo inicial.

Cualquier cambio en dicha estructura con respecto a la del modelo inicial debe ir acompañado de las razones que lo han aconsejado, que suelen ser ciertas características que en su momento no se conocía, o no se tuvieron en cuenta por ser un aspecto de mayor nivel de detalle.

Aunque en principio los requisitos -C afectan principalmente a la fase de análisis de requisitos y se utilizan para la creación de los requisitos -D (que evidentemente influyen directamente en la fase de diseño), puede darse el caso de que alguno de los requisitos del cliente tengan efecto directo sobre esta fase de diseño.

Implantación e Integración: también llamada codificación. Consiste en la codificación del software utilizando un lenguaje de programación siguiendo la estructura y comportamiento determinados en el diseño.

Revisiones del software que se aplican durante cada paso del desarrollo del mismo

Una vez que se ha creado una especificación y un diseño de un software, debe garantizarse su calidad. Las revisiones del software son un filtro para el proceso de ingeniería del software y se aplican en varios momentos del desarrollo. Sirven para detectar fallos

tanto en el análisis como en el diseño y la codificación, de manera que puedan ser eliminados cuanto antes. Es necesario partir de la idea de que todo el mundo comete errores, pero algunos de ellos se le pasan por alto más fácilmente al que los origina que a otras personas. Los errores cometidos en pasos anteriores se amplifican en el paso siguiente. Según las estadísticas, las actividades de diseño introducen entre el 50 y el 65 por 100 de todos los errores, por lo que cualquier técnica que permita detectar los errores en las fases iniciales del desarrollo del software será de gran utilidad.

Existen muchos tipos diferentes de revisiones dependiendo de qué se revise y el tipo de profesional que lo revise. Una de ellas son las revisiones técnicas formales o inspecciones, llevadas a cabo por ingenieros del software. Las revisiones técnicas formales son el filtro más efectivo desde el punto de vista de la garantía del software, ya que detectan el 75% de los errores cometidos en el diseño. Los objetivos de la revisión técnica formal son:

(a) Descubrir errores en la función, la lógica o la implantación de cualquier representación del software.

(b) Verificar que el software bajo revisión alcanza sus requisitos.

(c) Garantizar que el software ha sido representado de acuerdo con ciertos estándares predefinidos.

(d) Conseguir un software desarrollado de forma uniforme.

(e) Hacer que los proyectos sean más manejables.

Estrategia de prueba

La prueba del software es un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y la codificación. La prueba requiere que se descarten las ideas preconcebidas sobre la "corrección" del software que se acaba de desarrollar y se supere cualquier conflicto de intereses que aparezca cuando se detecten errores.

En un libro clásico sobre la prueba del software (Myers, 1979) se establecen una serie de reglas que pueden entenderse como objetivos de la prueba:

(a) La prueba es un proceso de ejecución de un programa con la intención de descubrir un error.

(b) Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.

(c) Una prueba tiene éxito si descubre un error no detectado hasta entonces.

Debido a la importancia de las pruebas para la calidad y a su dificultad, existen múltiples técnicas de prueba. En (Press m, 1995) se muestran algunas de estas técnicas de prueba.

Procedimiento que asegure un ajuste a los estándares de desarrollo del software

En la segunda parte de este artículo se muestran algunos de los estándares de calidad más utilizados hoy en día.

Gestión de configuraciones de software (control de la documentación del software y de los cambios realizados)

La gestión de configuraciones del software es una actividad "protectora" que se aplica a lo largo del proceso de ingeniería del software. Se trata de un conjunto de actividades de seguimiento y control que comienza al principio del proyecto de desarrollo del software y finaliza sólo una vez que el software queda fuera de circulación. Los elementos que componen toda la información producida se denominan configuración del software (programas, documentos que describen los programas y estructuras de datos). La elaboración de la documentación resulta muy costosa, por lo que es necesario intentar reducirla lo más posible y realizarla cuando los beneficios

que conlleve superen el coste de su realización.

Una de las principales amenazas para la calidad del software viene de una fuente aparentemente benigna: los cambios. El proceso de control de cambios contribuye directamente a la calidad del software. El control de cambios se aplica durante el desarrollo del software y, posteriormente, durante su mantenimiento. Ya que un cambio se puede producir en cualquier momento, las actividades de la gestión de configuraciones del software sirven para: (1) identificar el cambio; (2) controlar el cambio; (3) garantizar que el cambio se implementa adecuadamente; (4) informar del cambio a todos aquéllos a los que afecte.

En (Belin, 1998) se muestran de manera resumida las ideas principales de la gestión de la configuración.

Mecanismos de medida

La medición es una actividad fundamental para cualquier disciplina de ingeniería. Un objetivo importante de la garantía de calidad es seguir la pista a la calidad del software y evaluar el impacto de los cambios de metodología y de procedimiento que intentan mejorar la calidad del software. Para conseguirlo, se deben recolectar métricas del software, como se verá más adelante.

Registro y realización de informes

Son procedimientos para la recolección y divulgación de información de la garantía de calidad del software. Los resultados de las revisiones, auditorías, control de cambios, prueba y otras actividades de la garantía de calidad deben convertirse en una parte del registro histórico de un proyecto. Además, deben ser divulgados a la plantilla de desarrollo para que tenga conocimiento de ellos.

Conclusiones

En esta primera parte del artículo se han pesentado los factores que afectan a la calidad del software y cuáles son las tareas que hay que realizar para garantizarla. En la segunda parte del artículo se estudiarán con cierto detalle dos de estas tareas: los estándares y modelos de referencia y los mecanismos de medida. También se presentarán las restricciones que tiene algunas de estas técnicas y su uso en la actualidad. Finalmente, se presentarán las conclusiones generales del artículo.

Bibliografía

Belin, M. d. L. (1998). La gestion de configuration: points de repère. Revue de l'Electricité et de l'Electronique, 6(Juin), 58-61.

Brackett, J. W. (1990). Software Requirements (sei-cm-19-1.2): Canegie Mellon University - Software Engineering Institute.

Durán Toro, A., Bernárdez Jiménez, B., Corchuelo Gil, R., & Toro Bonilla, M. (2000). Ingeniería de Requisitos y Tecnología de Objetos. Novática, Ene/Feb, 15-20.

Dutta, S., Lee, M., & Van Wassenhove, L. (1999). Software Engineering in Europe: A study of best practices. IEEE Software, May/June, 82-90.

Fitzgerald, B., & O'Kane, T. (1999). A Longitudinal Study of Software Process Improvement. IEEE Software(May/June), 37-45.

IEEE. (1990). IEEE Standard Glossary of Software Engineering Terminology. IEEE/ANSI Standard 610.12-1990 : IEEE.

McCall, J. (1977). Factors in Software Quality : General Electric.

Myers, G. J. (1979). The Art of Software Testing: Wiley-Interscience.

Nistal Rosique, G. (1999). Presentación del monográfico de Calidad del Software. Novática, 137(En-Feb).

Pressman, R. S. (1995). Ingeniería del Software: Un enfoque práctico (Software Engineering. A Practitioner's Approach., Trans.). (Cuarta Edición ed.): McGraw-Hill/Interamérica de España, S.A.