Técnico en

DESARROLLO DE SOFTWARE

Aseguramiento de Calidad del Desarrollo de Software

Instituto Von Neumann

Aseguramiento de Calidad del Desarrollo de Software

Unidad I

1. Introducción

La calidad es el conjunto de propiedades inherentes a una entidad, que permiten juzgar su valor. Está cuantificada por el valor que se le da al conjunto de propiedades seleccionadas. De esta manera la calidad es subjetiva y, como dice James Bach [11], es circunstancial. Es subjetiva porque depende de los atributos elegidos para medirla y es circunstancial porque el conjunto de atributos elegidos puede variar en situaciones diferentes.

Cuando aplicamos el concepto de calidad al software, éste deja de ser subjetivo porque se determinan cuáles son los atributos de calidad del software. Pero no deja de ser accidental ya que, en ciertas situaciones, un determinado conjunto de características de calidad puede ser más importante que en ciertas otras.

Resumiendo, la calidad del software es medible y varía de un sistema a otro o de un programa a otro.

2. Calidad del software

Hablamos todo el tiempo de problemas relacionados con la calidad del software, pero no tenemos una definición precisa de lo que esto significa. Sin una definición clara, concisa y medible de lo que es la calidad del software, no podemos tomar buenas decisiones de negocio respecto del uso de los recursos, ni en qué áreas mejorar la calidad, ni qué herramientas y técnicas utilizar para mejorar la calidad.

Hay diferentes puntos de vista para definir calidad de software. Desde el punto de vista del cumplimiento de los requerimientos Roger Pressman define la calidad de software como:

"*El cumplimiento de los requerimientos funcionales y de performance explícitamente definidos, de los estándares de desarrollo explícitamente documentados y de las características implícitas esperadas del desarrollo de software profesional*." [1]

Desde el punto de vista del cliente o usuario Watts Humphrey dice:

"*El foco principal de cualquier definición de calidad de software debería ser las necesidades del cliente. Crosby [6] al igual que Pressman [1] define la calidad como conformidad con los requerimientos. Mientras uno puede discutir la diferencia entre requerimientos, necesidades y deseos, la definición de calidad debe considerar la perspectiva de los usuarios. Entonces las preguntas claves son ¿Quiénes son los usuarios?, ¿Qué es importante para ellos? Y ¿Cómo sus prioridades se relacionan con la manera en que se construye, empaqueta y se da soporte al producto*? [2]"

Al Davis define calidad del software como:

“La calidad no se trata de tener cero defectos o una mejora medible de la proporción de defectos, no se trata de tener los requerimientos documentados. No es más ni menos que satisfacer las necesidades del cliente (por más que las necesidades estén o no correctamente documentadas) [5]”

Finalmente, desde estas dos perspectivas el glosario de la IEEE para la ingeniería de software define la calidad del software como:

“El grado con el cual un sistema, componente o proceso cumple con los requerimientos y con las necesidades y expectativas del usuario. [4]”

Más allá de cómo definamos la calidad del software, para que la definición tenga sentido esta debe ser medible. Para poder controlar la calidad del software es necesario, ante todo, definir los parámetros, indicadores o criterios de medición, ya que, como bien plantea Tom De Marco, "no se puede controlar lo que no se puede medir".

Para poder identificar los costos y beneficios del software se definieron los atributos de calidad. La intención es separar el software en atributos que puedan ser medidos o cuantificados (en términos de costo beneficio). Ejemplos de estos atributos son confiabilidad, adaptabilidad, usabilidad y funcionalidad.

Para clasificar los atributos de calidad del software se definieron varios modelos, uno de ellos fue el modelo FURPS+. Este modelo fue desarrollado por Robert Grady y Deborah Caswell de Hewlett Packard

Bajo el acrónimo FURPS+, por sus siglas en inglés, se definen las siguientes características:

Sigla	Tipo de Requerimiento		Descripción
F	Funtional	***Funcional***	Características, capacidades y algunos aspectos de seguridad
U	Usability	***Facilidad de uso***	Factores humanos (interacción), ayuda, documentación
R	Reliability	***Reusabilidad***	Frecuencia de fallos, capacidad de recuperación de un fallo y grado de previsión
P	Perfomance	***Rendimiento***	Tiempos de respuesta, productividad, precisión, disponibilidad, uso de los recursos.
S	Supportability	***Soporte***	Adaptabilidad, facilidad de mantenimiento, Internacionalización, facilidad de configuración
+	Plus	***Implementación***	Limitación de recursos, lenguajes y herramientas hardware
		Interfaz	Restricciones impuestas para la interacción con sistemas externos (no es GUI)
		Operaciones	Gestión del sistema, pautas administrativas, puesta en marcha
		Empaquetamiento	Forma de distribución
		Legales	Licencia, derechos de autos, etc.

Tabla 1: Fuente: Adaptado de Craig Larman: “UML y Patrones” 2Ed. 2003.

El más (+) en el acrónimo FURPS+ nos permite especificar restricciones, incluyendo restricciones de diseño, implementación e interfaces.

La obtención de un software con calidad implica la utilización de metodologías o procedimientos estándares para el análisis, diseño, programación y prueba del

software que permitan uniformar la filosofía de trabajo, en aras de lograr una mayor confiabilidad, mantenibilidad y facilidad de prueba, a la vez que eleven la productividad, tanto para la labor de desarrollo como para el control de la calidad del software.

Cuando no se cumplen los estándares o procesos de la organización o del proyecto se dice que estamos frente a una no conformidad. Lo esperable es la ausencia de no conformidades (conformidad)

El costo de conformidad (calidad) no es despreciable, pero es menor que el costo de su alternativa (Costo de no conformidad). Crosby describe el costo de la no-conformidad como aquel costo en el que se incurre porque el producto o servicio no se desarrolló de forma apropiada la primera vez.

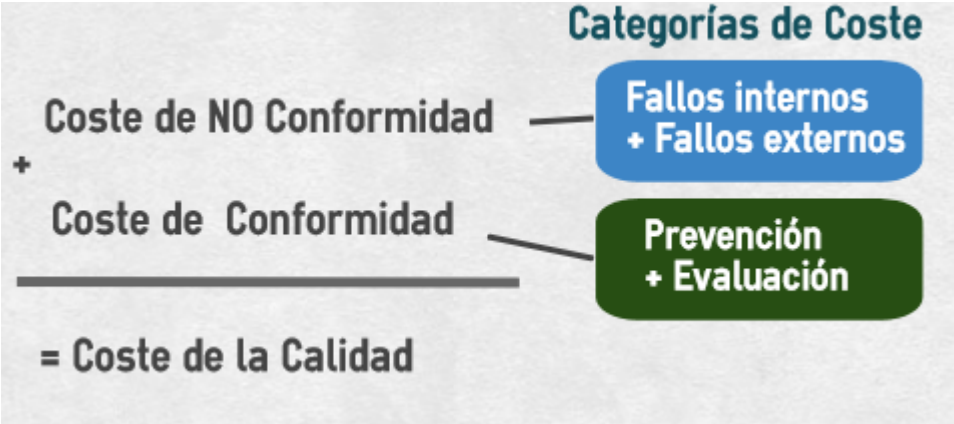

Figura 1: Ecuación para obtener el costo de la calidad

La calidad del software puede medirse después de elaborado el producto. Pero esto puede resultar muy costoso si se detectan problemas derivados de imperfecciones en el diseño o requerimientos, por lo que es imprescindible tener en cuenta tanto la

obtención de la calidad como su control durante todas las etapas del ciclo de vida del software.

En el siguiente cuadro Pressman muestra el costo relativo de corregir un error en las distintas etapas del ciclo de vida.

Fase	Requisitos	Diseño	Código	Prueba (desarrollo)	Prueba (sistema)	Explotación, producción
Multiplicador Coste	1	3-6	10	15-40	30-70	40-1000

Tabla 2: Costo de un error en las diferentes etapas del proceso.

Como se puede observar en la tabla , el costo relativo de corregir un error en producción es cientos de veces más que en la etapa de requisitos. El 50% del costo de corrección de defectos en las últimas fases se debe a defectos introducidos en las primeras.

Entonces uno de los propósitos que guían el aseguramiento de la calidad es que es mucho menos costoso corregir los problemas en sus fases iniciales que esperar hasta que un problema se manifieste a través de las quejas del usuario.

Los principios básicos del concepto de calidad del software son:

- No alcanza en pensar en calidad a la hora hacer las revisiones y pruebas, sino que debe ser una preocupación durante todo el ciclo de vida del software.
- Sólo se alcanza con la contribución de todas las personas involucradas.
- La calidad debe ser planificada y gestionada con eficacia.
- Dirigir esfuerzos a prevención de defectos.

- Reforzar los sistemas de detección y eliminación de defectos durante las primeras fases.
- La calidad es un parámetro importante del proyecto al mismo nivel que los plazos de entrega, costo y productividad.
- Es esencial la participación de la dirección, que ha de propiciar la calidad.

Hasta aquí, lo que define el modelo FURPS+ son los atributos de calidad de los productos de software, pero también los procesos que se siguen en su desarrollo cuentan con atributos de calidad como por ejemplo que estén bien definidos, que estén documentados, que sean practicados y medidos. La calidad depende de los procesos, debe estar orientada al cliente y es de mejora continua.

3. Aseguramiento de la calidad del software

3.1 Surgimiento de SQA (Software Quality Assurance)

En los años 50, el software comenzó a encontrar su camino dentro de los sistemas del DoD (del inglés Department of Defense of USA). Usualmente estos proyectos estaban muy alejados de la planificación, se pasaban del presupuesto y tenían muchos problemas técnicos. Frecuentemente no funcionaban como se esperaba y muchos proyectos eran cancelados antes de ser entregados. Durante este periodo los contratistas para el desarrollo de software a menudo hacían estimaciones muy optimistas sobre el estado del desarrollo del software. El DoD normalmente no era notificado de los problemas en la planificación, en la gestión del presupuesto y de problemas técnicos hasta muy avanzado el proyecto, cuando ya no eran capaces de entender los problemas ni de evaluar el impacto de éstos.

Para intentar resolver este problema se estableció la Verificación y Validación Independientes (IV&V del inglés Independent Verification and Validation), un proceso de ingeniería que empleaba metodologías rigurosas para evaluar la correctitud y calidad del software a lo largo de su ciclo de vida.

El primer software en usar IV&V fue el programa del misil atlas a finales de los años 50. Desde el proyecto atlas se ha recolectado mucha información que indica que los proyectos con IV&V se realizan o ejecutan mucho mejor que los proyectos sin IV&V. Con el tiempo el rol del IV&V se convirtió crítico.

La actividad que llamamos SQA evoluciona directamente de la Verificación y Validación Independientes (IV&V), muchas de las tareas que asociamos con SQA son originarias de IV&V.

Luego durante los años 70 la actividad de desarrollo de software comenzó a expandirse y las compañías de desarrollo de software fueron experimentando los mismos pobres resultados que las agencias gubernamentales (DoD, NASA etc.) en las décadas tempranas. Las compañías tenían dificultad para entregar el software dentro de los plazos, presupuesto y calidad planificados. Varios proyectos desarrollados entre 1980 y 1990 fueron desastrosos, muchos excedían ampliamente el presupuesto y la planificación o entregaban software de baja calidad que no se podía usar.

Durante los 80 esta experiencia se convirtió en lo que conocemos como crisis del software, el tiempo consumido en el mantenimiento excede el tiempo insumido en la construcción de nuevos productos de software.

Luego de la crisis del Software en los años 80, SQA evoluciono hacia una herramienta que las compañías de desarrollo de software utilizaban para identificar de forma temprana los problemas de calidad en el proceso de desarrollo. Mientras SQA era visto como un pequeño paso dentro del proceso del desarrollo del software, muchos jefes de proyectos vieron beneficios cuantificables a partir de integrar SQA dentro del proceso de desarrollo de software.

En los 90 varias compañías de software ya tenían funciones de SQA dentro de sus organizaciones.

3.2 Definición de SQA (Software Quality Assurance)

Al igual que ocurrió con la definición de calidad hay varios puntos de vista desde donde se puede definir el aseguramiento de la calidad del software.

Desde el punto de vista de la evidencia, la IEEE define el aseguramiento de la calidad como

"Una guía planificada y sistemática de todas las acciones necesarias para proveer la evidencia adecuada de que un producto cumple los requerimientos técnicos establecidos. Un conjunto de actividades diseñadas para evaluar el proceso por el cual un producto es desarrollado o construido." [4]

Daniel Galin define SQA como

"Un conjunto, sistemático y planificado, de acciones necesarias para proveer la evidencia adecuada de que el proceso de desarrollo o mantenimiento de un sistema de software cumple los requerimientos técnicos funcionales tan bien como los

requerimientos gerenciales para cumplir la planificación y operar dentro del presupuesto confinado" [7].

Desde el punto de vista de la visibilidad, el SEI define SQA como

"El aseguramiento de la calidad del software provee claro control del proceso que está siendo usado por el proyecto y del producto que se está construyendo." [8]

Desde el punto de vista del aseguramiento, Don Reifer define SQA como

"El aseguramiento de la calidad del software es el sistema de métodos y procedimientos usados para asegurar que el producto de software alcanza sus requerimientos. El sistema involucra la planificación, estimación y monitoreo de las actividades de desarrollo realizadas por otros." [9]

Desde el punto de vista de la capacidad de uso Schulmeyer y McManus definen SQA como

"Las actividades sistemáticas que proveen evidencia de la capacidad o disponibilidad de uso del producto de software total." [10]

Para certificar madurez de procesos, hay que evidenciar que uno aplica un cierto proceso y para esto se deben registrar las distintas actividades de tal proceso de desarrollo, como éste es el objetivo que persigue el software a desarrollar como parte de esta tesis, elegiremos la definición que da la IEEE desde el punto de vista de la generación de evidencia adecuada que muestre que se cumple con el proceso que se dice seguir y con los requerimientos establecidos.

3.3. SQA no es lo mismo que SQC (Software Quality Control)

Generalmente cuando le preguntamos a un profesional de sistemas que es lo que entiende por aseguramiento de la calidad del software, inmediatamente comienza a hablar de testing, algunos de ellos incluyen a la validación y verificación y luego empiezan a hablar de revisiones, las cuales son sólo extensiones del testing. Es decir, a menudo hay una confusión entre SQA y el testing (el cual actualmente forma parte del área de control de calidad del software SQC).

Haciendo sólo testing y revisiones no aseguramos la calidad de los productos, sino aseguramos el cumplimiento de especificaciones tanto funcionales como técnicas. En el desarrollo de software la diferencia entre SQC y SQA no está clara y estos términos a menudo se confunden, SQA se encarga de controlar el cumplimiento del proceso, mientras que SQC son aquellas acciones del aseguramiento de la calidad que proporcionan un medio para controlar y medir las características de un elemento, proceso o facilidad respecto a los requisitos establecidos.

La siguiente tabla expone sintéticamente las diferencias entre control de calidad y aseguramiento de la calidad.

Control de calidad	Aseguramiento de la calidad
Detecta problemas en los productos de trabajo.	Asegura la adherencia a los procesos, estándares y planes.
Verifica que los productos de trabajo cumplan con los estándares de calidad especificados en el plan de proyecto.	Evalúa que los procesos, planes y estándares utilizados en el proyecto cumplan con los estándares organizacionales.
Revisa el contenido del producto	Revisa procesos

Tabla 3: Control de calidad vs. Aseguramiento de la calidad

En conclusión, el rol del SQA es auditar que los distintos equipos de la organización, inclusive el de SQC siguen los procedimientos, estándares y procesos establecidos. El equipo de SQA debería establecer métricas para medir la efectividad del proceso. Como complemento el rol de SQC es tomar una actitud activa de verificación y validación del resultado o salida del proceso implementado.

3.4. Funciones generales del SQA

Describir los diferentes roles que puede jugar el equipo de SQA en una organización nos dará una visión clara de las funciones que puede llevar a cabo.

“Como policía del proceso”: el trabajo del equipo de SQA es asegurar que el desarrollo sigue el proceso establecido. Entre sus funciones en este rol se encuentran:

- Auditar los productos del trabajo para identificar deficiencias.
- Determinar el cumplimiento del plan de desarrollo del proyecto y del proceso de desarrollo de software. o Juzgar el proceso y no el producto.

“Como abogado del cliente”: el trabajo del equipo de SQA es representar al cliente. Entre sus funciones en este rol se encuentran:

- Identificar la funcionalidad que al cliente le gustaría encontrar.
- Ayudar a la organización a sensibilizarse con las necesidades del cliente. o Actuar como un cliente de prueba para obtener una alta satisfacción del cliente.

“Como analista” el trabajo del equipo de SQA es recabar información. Entre sus funciones en este rol se encuentran:

- Juntar muchos datos sobre todos los aspectos del producto y del proceso. o Con esta información ayudar a mejorar los procesos y los productos.

"Como proveedor de información" el trabajo del equipo de SQA es revisar qué es lo que está hecho y decir cuáles objetivos técnicos realmente están cumplidos para que la gerencia pueda tomar mejores decisiones de negocios. Entre sus funciones en este rol se encuentran:

- Proveer información técnica objetiva para que la gerencia pueda usarla para tomar mejores decisiones.
- Proveer información apropiada de las clases de productos y de los riesgos asociados con estos. o Concentrarse más en la reducción de los riesgos que en el cumplimiento del proceso.

"Como responsable de la elaboración del proceso" el trabajo del equipo de SQA es participar en la definición de los planes, procesos, estándares y procedimientos para asegurar que se ajustan a las necesidades del proyecto y que pueden ser usados para realizar las evaluaciones de QA y cumplir los requerimientos del proyecto y las políticas de la organización. Para cumplir este rol el aseguramiento de la calidad debería comenzar en las fases tempranas del proyecto.

Aquí conviene aclarar que no necesariamente las personas que definen la metodología a seguir pertenecen al equipo de QA. Definir la metodología puede llegar a ser o no una actividad del equipo de QA. Una estructura posible en el proceso de mejora del software puede ser contar con un SEPG (Software Engineering Process Group)

totalmente independiente del equipo de QA, encargado de definir la metodología mientras que el equipo de QA se limita a verificar que se cumpla dicha metodología.

3.5. Consideraciones

Para ser efectivo, el equipo que realiza SQA debe ser independiente de la organización de desarrollo. Aunque tener un grupo de auditoría independiente es difícil de aplicar en organizaciones chicas donde hay pequeños ambientes de desarrollos. Pero si la organización es madura y tiene una cultura orientada a la calidad, la función de SQA puede estar embebida en el proceso. Cuando el equipo de SQA esta embebido en el proceso, se deben resolver varios inconvenientes para garantizar la objetividad:

- Todo aquel que realice actividades de aseguramiento de la calidad debería estar entrenado en el aseguramiento de la calidad. o Las actividades de aseguramiento de la calidad realizadas para un producto deberían ser separadas de aquellas involucradas en el desarrollo y mantenimiento de este.
- Debe estar disponible un canal de comunicación independiente en el nivel apropiado de la gerencia para poder escalar las no conformidades cuando sea necesario.

El equipo de SQA provee a la gerencia de información fehaciente, objetiva en el momento adecuado. La clave aquí está en que el grupo de SQA provee a la gerencia de información técnica objetiva. La gerencia necesita ver a la gente de SQA como una fuente de información significativa que puede ayudarla a tomar decisiones difíciles. La Gerencia usa esta información para tomar decisiones de negocio apropiadas.

La objetividad en la evaluación de calidad de los procesos y productos es crítica para el éxito del proyecto. La objetividad se logra con independencia del equipo de SQA y sentido común o criterio.

Hay diferentes maneras de realizar evaluaciones objetivas, entre las que se incluyen:

- Auditorías formales realizadas por un área de SQA independiente de la organización.
- Revisiones de a pares que pueden ser realizadas con distintos niveles de formalidad.
- Revisiones rigurosas en el lugar de desarrollo.
- Revisiones distribuidas y comentarios del producto.

Teniendo en cuenta estas consideraciones podemos decir que la tarea del equipo de SQA es un conjunto planificado de tareas, actividades y acciones ejecutadas independientemente de la organización que desarrolla software, que provee a la gerencia del proyecto información fehaciente en un momento preciso que puede ser usada para tomar decisiones de negocio apropiadas.

Referencias

- [1] Roger Pressman - Software Engineering: A Practitioner's Approach
- [2] Humphrey, W., A Discipline for Software Engineering, Addison-Wesley, 1995.
- [4] IEEE Standard Glossary of Software Engineering Terminology, IEEE Std 610.12- 1990, 1990
- [5] Eickelman, N. and Hayes, J. eds., “New Year’s Resolutions for Software Quality”, IEEE Software, Jan-Feb 2004, pp. 12-13

- [6] Philip B. Cosby - Quality is Still Free: Making Quality Certain in Uncertain Times
- [7] Software Quality Assurance: From Theory to Implementation - Daniel Galin, Ruppin Academic Center, Israel
- [9] Donal J. Reifer - Software Maintenance Success Recipes
- [10] Software Quality Assurance, 3rd ed., G. G. Schulmeyer and J. I. McManus
- [11] James Bach - Lessons Learned in Software Testing

Descargo de responsabilidad

La información contenida en este documento descargable en formato PDF o PPT es un reflejo del material virtual presentado en la versión online del curso. Por lo tanto, su contenido, gráficos, links de consulta, acotaciones y comentarios son responsabilidad exclusiva de su(s) respectivo(s) autor(es) por lo que su contenido no compromete al área de e-Learning del Departamento GES o al programa académico al que pertenece.

El área de e-Learning no asume ninguna responsabilidad por la actualidad, exactitud, obligaciones de derechos de autor, integridad o calidad de los contenidos proporcionados y se aclara que la utilización de este descargable se encuentra limitada de manera expresa para los propósitos educacionales del curso.