UNIVERSIDAD DE LA CUENCA DEL PLATA

Ingeniería en Sistemas de Información

Actividad Áulica 27

Paradigmas de Programación III

Profesor: Mg. Agustin Encina

Carrera: Ingeniería en Sistemas de Información.

Cátedra: Paradigmas de Programación III

Año: Tercero.

Comisión: “A” (única).

Sede: Posadas.

Alumnos:

- Becerra, Tobias DNI: 46.084.038

Introducción

La calidad del software es un pilar fundamental dentro del desarrollo de sistemas actuales. Un producto tecnológico de buena calidad no solo opera sin fallas, sino que también garantiza estabilidad, eficiencia y una experiencia acorde a lo que espera el usuario. Por ello, la calidad no debe entenderse únicamente como la ausencia de errores técnicos, sino como una perspectiva completa que integra funcionamiento adecuado, buen desempeño, seguridad y satisfacción del usuario.

Las normas internacionales —como la familia ISO 25000 para evaluar la calidad y la ISO 38500 orientada al gobierno de TI— proporcionan marcos formales para optimizar procesos y asegurar la calidad del software. Asimismo, las diversas prácticas de testing ofrecen herramientas para comprobar que el sistema se comporte de manera correcta, mientras que los principios de la arquitectura SOA permiten construir soluciones escalables, modulares y fácilmente reutilizables.

En conjunto, todos estos componentes conforman un enfoque integral que favorece el desarrollo de productos confiables y sostenibles a lo largo del tiempo.

Calidad del Software

Concepto de Calidad de Software

La calidad del software puede comprenderse como el grado en que un sistema cumple los requisitos funcionales y no funcionales, proporcionando una experiencia de uso segura, estable y satisfactoria. Entre sus dimensiones más relevantes se encuentran:

- **Satisfacción del usuario:** el software debe resolver las necesidades reales del usuario final y ofrecer una interacción clara y eficiente.
- **Fiabilidad y estabilidad:** un sistema confiable minimiza errores, caídas y comportamientos inesperados.
- **Seguridad:** protege los datos y evita vulnerabilidades.
- **Eficiencia:** optimiza tiempos de respuesta y el uso de recursos.
- **Usabilidad:** permite que el usuario comprenda y utilice el sistema sin dificultades.

Un ejemplo ilustrativo fue el colapso repetido de servicios bancarios digitales durante períodos de alta demanda, lo que afectó la percepción de confiabilidad de los usuarios y evidenció la ausencia de pruebas de carga adecuadas.

Normas y Modelos de la Calidad

Las normas ISO/IEC 25000 e ISO 25010 ofrecen un marco estructurado que permite evaluar la calidad del software a partir de características cuantificables. Estas dimensiones permiten estandarizar el análisis del producto y generar indicadores medibles.

Dimensión	Descripción	Ejemplo
Funcionalidad	Evalúa si el sistema cumple con las funciones previstas.	Un sistema de facturación que emite comprobantes correctos.
Fiabilidad	Mide la estabilidad y consistencia del software en diversas condiciones.	Una aplicación bancaria estable incluso en horarios de pico.
Usabilidad	Determina la facilidad de uso para el usuario final.	Interfaces intuitivas con flujos claros.
Eficiencia	Analiza tiempos de respuesta y rendimiento del sistema.	Una web que carga en menos de dos segundos.
Mantenibilidad	Evalúa la facilidad para modificar, corregir o ampliar el software.	Código modular y organizado.
Portabilidad	Examina la capacidad del sistema para ejecutarse en entornos distintos.	Aplicaciones compatibles con Windows, Linux y macOS.

Testing y Tipos de Pruebas de Calidad de Software

El testing constituye la etapa destinada a verificar que cada componente del software funcione correctamente y cumpla con los requisitos definidos. Las pruebas permiten identificar defectos, prevenir errores futuros y mejorar la confiabilidad general del sistema.

Tipo de prueba	Momento	Objetivo	Ejemplo
Unitarias	Durante el desarrollo	Validar funciones específicas y aisladas.	Probar operaciones matemáticas.
Integración	Tras unir módulos	Verificar la comunicación entre componentes.	Probar login + base de datos.
Pruebas de sistema	Sistema completo	Validar el funcionamiento general.	Flujo completo de compra online.

Tipo de prueba	Momento	Objetivo	Ejemplo
Aceptación	Etapa final	Confirmar que el software cumple los requisitos del cliente.	Validación de reportes por parte del usuario.
Rendimiento	Antes del despliegue	Evaluar tiempos de respuesta y carga soportada.	Cuántos usuarios simultáneos soporta una web.
Seguridad	Entorno pre-productivo	Detectar vulnerabilidades.	Pruebas de SQL Injection.
Accesibilidad	Previo a producción	Confirmar que el sistema es accesible para todos los usuarios.	Navegación por teclado y contraste adecuado.

El caso del Boeing 737 MAX evidenció la importancia de realizar pruebas rigurosas: deficiencias en los sistemas de control y la ausencia de validaciones críticas derivaron en uno de los mayores fallos tecnológicos de los últimos años.

Métricas y Evaluación del Software

Una métrica es un valor numérico que permite medir de forma objetiva una característica del software, facilitando comparaciones, mejoras y la identificación de puntos críticos.

Métricas comunes en sistemas web

- **Tiempo de respuesta (latencia):** cuánto tarda el sistema en responder.
- **Throughput:** cantidad de transacciones o solicitudes procesadas por segundo.
- **Uso de recursos:** consumo de CPU, memoria o ancho de banda.

Estas métricas permiten evaluar la capacidad operativa del sistema y tomar decisiones informadas sobre su optimización.

Arquitectura SOA e ISO 38500

Concepto de SOA

La Arquitectura Orientada a Servicios (SOA) define un modelo de diseño donde las funcionalidades del sistema se presentan como servicios independientes, reutilizables y bien estructurados. Cada servicio cumple una función específica y puede interactuar con otros de manera controlada y estandarizada.

Principios fundamentales

- **Contrato estandarizado:** cada servicio expone interfaces claras y documentadas.
- **Bajo acoplamiento:** minimiza dependencias entre servicios.
- **Reutilización:** los servicios pueden utilizarse en distintos contextos.
- **Abstracción:** los detalles internos del servicio no se exponen al exterior.
- **Autonomía:** cada servicio controla su propia lógica.
- **Ausencia de estado:** el servicio no depende de información persistente entre solicitudes.
- **Composición:** servicios simples pueden unirse para formar soluciones más complejas.

Impacto de los principios en la calidad

Principio	Impacto
Contrato estandarizado	Reduce errores de integración y mejora la interoperabilidad.
Bajo acoplamiento	Facilita el mantenimiento y reemplazo de componentes.
Reutilización	Agiliza el desarrollo y aumenta la consistencia interna.
Abstracción	Disminuye la complejidad para quienes consumen el servicio.
Autonomía	Incrementa la confiabilidad de cada módulo.
Ausencia de estado	Favorece la escalabilidad horizontal.
Composición	Permite crear soluciones flexibles a partir de bloques simples.

Ejemplo práctico

Un sistema de comercio electrónico puede implementar servicios independientes para pagos, catálogo de productos, envíos y notificaciones. Esta separación permite actualizar cada componente sin afectar a los demás y reutilizarlos en múltiples plataformas.

Relación con ISO 38500

La ISO 38500 establece lineamientos para el gobierno corporativo de TI, asegurando que las decisiones tecnológicas —incluida la adopción de SOA— estén alineadas con la estrategia organizacional, los recursos disponibles y las necesidades del negocio.

Cuadro Integrador

Eje	Conceptos Clave	Contribución a la Calidad	Ejemplo
Modelos de calidad (ISO 25000)	SQuaRE, ISO 25010, características de calidad	Ofrecen un marco estandarizado para evaluar el software	Medición de tiempos de respuesta y estabilidad en una web.
Testing y métricas	Pruebas unitarias, integración, aceptación, seguridad y rendimiento; métricas de performance	Reducen fallas, previenen regresiones y mejoran confiabilidad	Pruebas de carga antes de un evento de alta demanda.
Arquitectura SOA	Bajo acoplamiento, autonomía, reutilización	Permite crear sistemas escalables y mantenibles	Servicios independientes para pagos, inventario y usuarios.
Gobierno de TI (ISO 38500)	Responsabilidad, estrategia, rendimiento	Asegura que la tecnología apoye objetivos del negocio	Inversión en arquitectura escalable y métricas de calidad.

Conclusiones

La calidad del software, las pruebas, las métricas y la arquitectura SOA forman un marco integral indispensable para el desarrollo tecnológico actual. Un sistema de calidad satisface los requisitos del usuario; las pruebas verifican su comportamiento adecuado; las métricas permiten medir su desempeño; y la arquitectura SOA facilita la creación de soluciones adaptables, modulares y escalables.

Asimismo, las normas ISO aportan un soporte formal que orienta a las organizaciones hacia procesos más eficientes, claros y uniformes. En conjunto, todos estos componentes sostienen un proceso continuo de mejora, en el que cada decisión técnica aporta a la construcción de un producto más sólido, confiable y alineado con las demandas del entorno empresarial moderno.