

Paradigmas y Lenguajes de Programación III

CARRERA: Ingeniería en Sistemas de Información

MATERIA: Paradigmas y Lenguajes de Programación III

COMISIÓN: "A"

PROFESOR: Mgter. Ing. Agustín Encina

ESTUDIANTE: Luciana Nadine Rojas

Consigna 1: ¿Qué relación existe entre calidad del software y satisfacción del usuario?

La calidad del software está directamente relacionada con la satisfacción del usuario, ya que un software de alta calidad cumple con las expectativas funcionales, es confiable, eficiente y fácil de usar. Cuando el software presenta fallas, lentitud o complejidad innecesaria, genera frustración y disminuye la confianza del usuario en la aplicación y en la organización que la desarrolló.

Consigna 2: ¿Qué dimensiones de la calidad menciona el video?

- Funcionalidad: Capacidad del software para cumplir con las funciones especificadas
- Confiabilidad: Capacidad de mantener el rendimiento bajo condiciones específicas
- Usabilidad: Facilidad de uso y aprendizaje
- Eficiencia: Rendimiento y uso optimizado de recurso
- Mantenibilidad: Facilidad para realizar modificaciones y correcciones
- Portabilidad: Capacidad para ser usado en diferentes entornos

Consigna 3: Ejemplo real de falla de calidad que afectó la confianza del usuario

Caso: Fallo en la aplicación móvil de Banco Galicia (2023)

- Falla: Usuarios no podían acceder a sus cuentas durante varias horas
- Impacto: Imposibilidad de realizar transferencias, pagos y consultas
- Consecuencia: Pérdida de confianza masiva, quejas en redes sociales, migración de clientes a otras entidades
- Causa raíz: Actualización con errores de código y falta de testing exhaustivo

ETAPA 2 – NORMAS Y MODELOS DE CALIDAD

Consigna 1: Componentes principales del modelo SQuaRE

- ISO 2500n: Guías de gestión de calidad
- ISO 2501n: Modelo de calidad
- ISO 2502n: Métricas de calidad

- ISO 2503n: Requisitos de calidad
- ISO 2504n: Evaluación de calidad

Consigna 2: Cuadro de dimensiones de calidad

Dimensión	Descripción breve	Ejemplo en aplicación en software
Funcionalidad	Grado en que el software satisface necesidades explícitas e implícitas	Sistema de ventas online que procesa correctamente pagos, descuentos e inventario
Fiabilidad	Capacidad de mantener nivel de desempeño bajo condiciones específicas	Aplicación bancaria que funciona 24/7 sin caídas y recupera transacciones tras fallos
Usabilidad	Facilidad de aprendizaje, uso y satisfacción del usuario	Interfaz intuitiva de Netflix con navegación simple y recomendaciones personalizadas
Eficiencia	Rendimiento relativo a cantidad de recursos utilizados	WordPress que maneja 1000 visitas simultáneas con respuesta en menos de 2 segundos
Mantenibilidad	Facilidad para ser modificado (correcciones, mejoras o adaptación)	Código modular de React que permite actualizar componentes sin afectar toda la aplicación
Portabilidad	Capacidad para ser transferido de un entorno a otro	Aplicación Java que funciona igual en Windows, Linux y macOS sin modificaciones

Reflexión: ¿Qué importancia tiene la medición de la calidad en el proceso de desarrollo?

La medición de la calidad es fundamental porque permite:

- Tomar decisiones basadas en datos en lugar de suposiciones
- Detectar problemas tempranamente, reduciendo costos de corrección
- Establecer benchmarks para mejora continua
- Garantizar que el producto final cumple con expectativas de stakeholders
- Prevenir riesgos que podrían afectar la reputación y operatividad

ETAPA 3 – TESTING Y TIPOS DE PRUEBAS

Cuadro comparativo de tipos de test

Tipo de Test	Momento del ciclo	Objetivo principal	Ejemplo práctico
Unidad	Durante desarrollo	Verificar funcionamiento de componentes individuales	Probar función calcularTotal() con diferentes inputs
Integración	Tras pruebas de unidad	Validar interacción entre módulos conectados	Probar que módulo "usuarios" se comunica correctamente con "pedidos"
Sistema	Al completar integración	Validar sistema completo según requisitos	Probar flujo completo: registro → login → compra → confirmación
Aceptación	Antes del release	Validar que cumple necesidades del cliente	Cliente prueba aplicación en entorno similar a producción
Rendimiento	Post-integracion/Pre-producción	Evaluar comportamiento bajo carga	Simular 1000 usuarios concurrentes en e-commerce
Seguridad	Durante y post-desarrollo	Identificar vulnerabilidades	Test de inyección SQL y XSS en formularios de login
Accesibilidad	Durante y post-desarrollo	Garantizar acceso a usuarios con discapacidades	Verificar contraste, lectores pantalla, navegación por teclado

Caso real de falla por falta de testing

Caso: Fallo en actualización de software de aviones Boeing 737 MAX

- Falla: Sistema MCAS (Maneuvering Characteristics Augmentation System) activándose erróneamente
- Consecuencia: Dos accidentes fatales (Lion Air 2018 y Ethiopian Airlines 2019)
- Causa raíz: Testing insuficiente del software de control de vuelo
- Problemas detectados:
- No se probaron escenarios extremos de datos de sensores
- Falta de pruebas de integración entre hardware y software
- Omitieron pruebas de seguridad críticas para reducir tiempos
- Impacto: 346 víctimas fatales, grounding mundial de flota, pérdidas billonarias

ETAPA 4 – MÉTRICAS Y EVALUACIÓN DE PERFORMANCE

Consigna 1: ¿Qué es una métrica de calidad?

Una métrica de calidad es un indicador cuantificable que permite medir atributos específicos del software para evaluar su nivel de calidad objetivamente. Proporciona datos concretos sobre el comportamiento, rendimiento y características del sistema.

Consigna 2: 3 métricas para evaluar performance de sistema web

1. Tiempo de respuesta: Segundos para cargar página completa (objetivo: <3s)
2. Throughput: Peticiones por segundo que el servidor puede manejar (ej: 1000 req/seg)
3. Tasa de error: Porcentaje de solicitudes fallidas vs totales (objetivo: <1%)

Consigna 3: Reflexión sobre medición cuantitativa

La medición cuantitativa es esencial porque:

- Elimina subjetividad en evaluaciones de calidad
- Permite comparación objetiva entre versiones y alternativas
- Facilita identificación de tendencias y patrones problemáticos
- Proporciona base para mejora continua mediante métricas específicas
- Ayuda en asignación de recursos priorizando áreas críticas

ETAPA 5 – ARQUITECTURA SOA Y NORMA ISO/IEC 38500

Consigna 1: Principios de SOA

Consigna 2: Cómo cada principio mejora calidad e interoperabilidad

Principio SOA	Mejora en Calidad	Mejora en Interoperabilidad
Contrato estandarizado	Interfaces consistentes y documentadas	Servicios pueden comunicarse sin conocimiento interno
Bajo acoplamiento	Cambios en un servicio no afectan otros	Servicios evolucionan independientemente
Reutilización	Reduce duplicación y errores	Mismos servicios usados en múltiples sistemas
Abstracción	Oculta complejidad interna	Interfaz simple para consumo externo
Autonomía	Servicios controlan propia lógica y datos	Operación independiente de otros servicios
Ausencia de estado	Mejora escalabilidad y rendimiento	No depende de estado de sesiones anteriores
Composición	Crea servicios complejos a partir de simples	Integración flexible de funcionalidades

Consigna 3: Ejemplo real de aplicación SOA

Empresa: Amazon Web Services (AWS)

- Situación: Necesitaban escalar infraestructura para múltiples servicios cloud
- Solución SOA: Implementaron arquitectura basada en microservicios
- Aplicación:
- Cada servicio (EC2, S3, Lambda) es independiente
- Comunicación vía APIs REST estandarizadas
- Bajo acoplamiento entre servicios

- Reutilización de servicios comunes (autenticación, billing)
- Resultado: Escalabilidad masiva, lanzamiento rápido de nuevos servicios, alta disponibilidad

Consigna 4: Conexión con norma ISO 38500

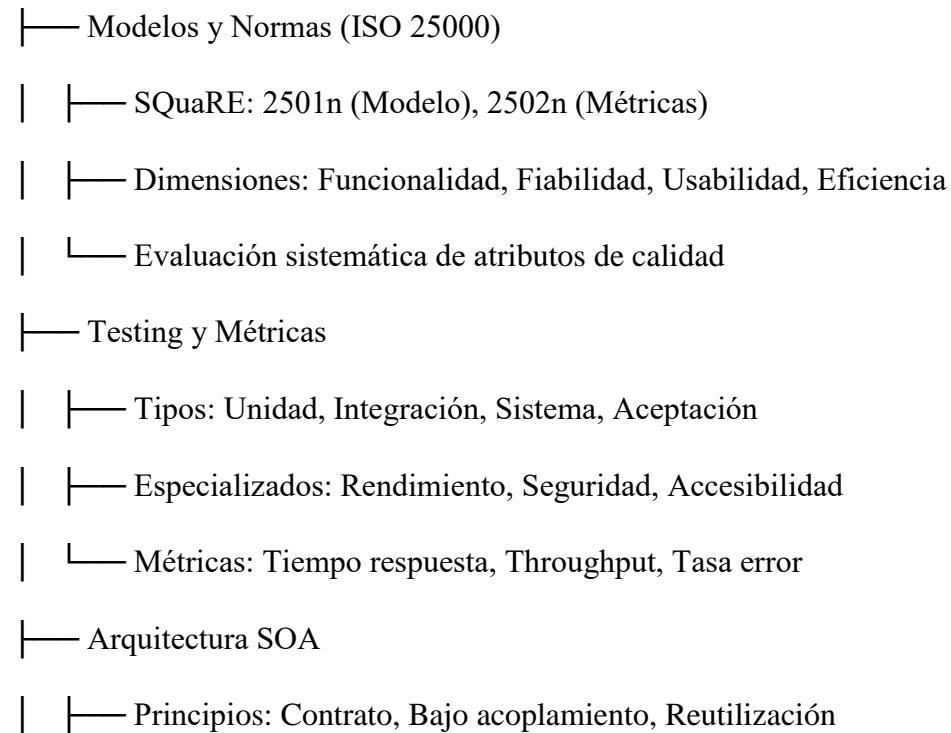
La norma ISO 38500 contribuye al gobierno de TI mediante:

- Marco para dirección efectiva de recursos tecnológicos
- Principios de gobierno: Responsabilidad, estrategia, adquisición, rendimiento, conformidad y comportamiento humano
- Alineación de TI con objetivos de negocio
- Gestión de riesgos tecnológicos
- Evaluación continua de desempeño de TI

ETAPA 6 – CIERRE Y REFLEXIÓN

Mapa Conceptual Integrador

CALIDAD DEL SOFTWARE



- | └— Beneficios: Interoperabilidad, Escalabilidad, Mantenibilidad
- | └— Implementación: Microservicios, APIs, Composición
- └— Gobierno de TI (ISO 38500)
 - └— Dirección estratégica de TI
 - └— Alineación con objetivos negocio
 - └— Gestión de riesgos y compliance

Conclusiones

¿Qué relación hay entre calidad, testing y arquitectura en el desarrollo tecnológico?

La calidad, testing y arquitectura forman un triángulo interdependiente en el desarrollo tecnológico. La arquitectura (especialmente SOA) establece los cimientos para sistemas mantenibles y escalables. El testing valida que estos sistemas funcionan correctamente según especificaciones. La calidad es el resultado de una buena arquitectura combinada con testing exhaustivo. Sin arquitectura sólida, el testing se vuelve complejo; sin testing, la calidad no puede verificarse; sin enfoque en calidad, la arquitectura y testing pierden propósito.

¿Cómo pueden estas normas fomentar la innovación y la confianza en los sistemas?

Las normas como ISO 25000 y 38500 fomentan la innovación al proporcionar:

- Marcos estructurados que permiten experimentación controlada
- Estándares comunes que facilitan integración de nuevas tecnologías
- Gestión de riesgos que permite asumir desafíos innovadores con seguridad
- Métricas objetivas para evaluar nuevas soluciones

La confianza se construye mediante:

- Transparencia en procesos de desarrollo y evaluación
- Consistencia en calidad entregada a usuarios
- Cumplimiento de estándares reconocidos internacionalmente
- Capacidad de evolución sin comprometer estabilidad