

# RESUMENES CALIDAD DEL SOFTWARE.

## TEMA 11 POR QUÉ CALIDAD DEL SOFTWARE.

La calidad está en el proceso, en las personas y en el producto.

- La calidad del producto depende de las personas y el proceso, pero puedo mirar la calidad del producto sin mirar a las personas/proceso

Puntos de Calidad Software:

- Que algo funciona no implica que este bien construido. Una prueba de caja negra no es suficiente.
- No gestionar la calidad genera deuda técnica. (Sonar y kivanc para este punto)
- La mala calidad siempre la paga alguien.
- Las certificaciones no son suficiente. Calidad del producto.
- Ojo al externalizar, relación cliente - proveedor (calidad producto)
- Control de versiones, jenkins ...

Regla del 80% - 20%: El 80% de los problemas vienen del 20% del código. (Soluciones a Búndes infinitos, líneas 1000., Copy/Paste

- Modularidad / Subprogramas / Refactorización.
- Pair Programming
- Testing de Regresión.
- Análisis automatizado / Sprints.

La materiabilidad del código pasa por, analizar (ser comprensible) y cambiar (Poder modificarlo)

- Tener pruebas unitarias: Más rápidas y localizan los errores más concretamente (Unitario = solo 1 objeto)

Mejores que integración y funcional

Unitario → Principio de Inversión de la dependencia.

A veces interesa la deuda técnica apostar (Beneficio Económico)

- Lo haces mal y rápido → Te prefieren
- Fidelidad Clientes → Proyecto chupasangre.
- Solo I puede tocar el código
- Competitivo
- Síndrome Ventana rota (Mal código lleva a mal código)

Causas de las malas decisiones

- Triángulo Funcionalidad, Tiempo y Costo.
- Causas naturales: Entropía (El código tiende a degradarse)
- Vivir en el pasado → Industrialización.

Una Certificación (CHAI) no certifica calidad / Capability Maturity Model

## TEMA 2. MODELOS Y NORMAS. SIX SIGMA.

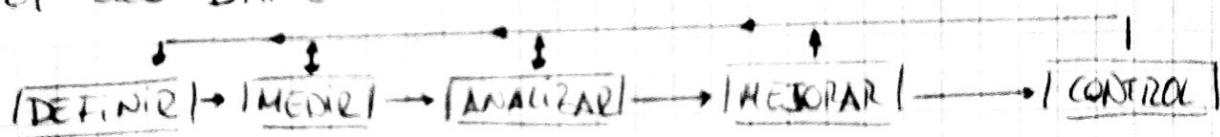
### 1. Qué es Six Sigma.

- Metodología de gestión de la calidad. (lenguaje negocios)
- Paradigma que transforma la metodología científica en cada organización.
- Six Sigma se basa en ciclo DMAIC (5 fases)
  - Definición, Medición, Análisis, Mejora, Control.
- Aplica el método científico para mejorar procesos
  - consecución de un fin.

### 2. Six Sigma y el Método científico.

- Pasos como método científico.
- Traduce el lenguaje científico en una forma sencilla.
  - Analizar entradas para saber qué pide el cliente
    - COMUNICACIÓN BIDIRECCIONAL.

### 3. El Ciclo DMAIC



a) **DEFINIR**: Se determina si six sigma es adecuado.

• Clave → Acta de constitución del proyecto (Project Charter)

- Alcance / Objetivos y Participantes.

- Roles y Responsabilidades

- Identifica objetivos y stakeholders

- Determina jerarquía del proyecto

• lista de características críticas para la calidad (CTQ)

Critical to Quality

- Herramientas utilizadas: Mapa del proceso y análisis de función de pérdidas.
  - CQA (Cost of Quality): Coste asociado a un producto con mala calidad
  - TQC (Total Quality Cost) Coste del producto total.
- b) MEDICIÓN: Extraer toda la información posible del proceso.
- Crear un mapa de proceso detallado, recopilar datos y resumir la información.
    - Mapa de proceso: Representación visual, Identificar requerimientos del proceso.
  - MSA (Measurement System Analysis): Análisis del sistema de medida.
  - Análisis de Pareto → Priorizar causas de defectos (El 20% de los problemas vienen del 80% del código)

- c) ANÁLISIS: Identificar causas primarias del problema. (Validar con Datos)
- Mapa de Proceso, Datos de las otras fases se deben usar para determinar causas primarias
  - Análisis Estadístico. → Gráficos (Elegir variables, Tipo de var, ...)

- d) MEJORA: Formar un equipo de mejora.
- Determinar una solución → Brainstorming
  - Involucrar personal del proceso.
  - Crear un mapa de estados futuros (Páginas y simulación)

## e) CONTROL : Consolidar ganancias

- Control de variables de entrada, otras final de inspección.
- Reconocer éxitos → recompensas.
- Poka - Yoke : Evitar influencia externa.
- Gráficas de control → monitoriza rendimiento.

## GESTIÓN DE CALIDAD.

### 1. Introducción.

- Mala Calidad = Perdida Clientes y genera competidores.
- Calidad → antepone necesidades del Cliente.

### 2. Definición calidad.

- No existe solo una definición. (Cambia con el tiempo)
- Conformidad con especificaciones → cumple objetivos.
- Valor por el precio pagado

### 3. Costes de la Calidad.

- Costes para alcanzar la calidad (Prevención y evaluación)
- Costes de consecuencia de mala calidad.

### 4. Evolución de la gestión de calidad.

- Calidad como ventaja competitiva.
- Después → naturaleza estadística.
- TQM (Gestión de la calidad total)

### 5. Filosofía de Calidad total.

- Se le da el valor que pide el cliente
- Mejora continua → kaizen (Estudiar al competidor)
- Diagramas de distintos tipos

## 6. Normas de Calidad.

- Proveedores cumplen normas, que no se necesita comprobar materiales
- Normas universales de calidad → ISO
  - ISO 9000:2000 : Terminología de normas
  - ISO 9001:2000 : Cumplimiento requisitos clientes
  - ISO 9004:2000 : Pautas de sist. gestión calidad
  - Hay que renovar la certificación.

## 7. Porque faltan los esfuerzos TQM

- Carencia de cultura de calidad
- Carencia apoyo ejecutivos

## TERMINOLOGÍA.

- Certificación: Empresa externa hace una evaluación y comprueba si cumple.
- Acreditación: Empresas que hacen las certificaciones (auditoría) deben estar acreditadas por otras.
- Norma: Requisito a cumplir.
- Auditoría: Prueba a pasar.

### - Modelos para el proceso.

- ESCH - CL/SP: Mejorar comunicación empresa y desarrollador.
- CHMI: Modelo documentación madurez.
- ISO 2000: Genérica.
- ISO 15504 / 12207: Evalua procesos Organización.

### - Modelos para el producto

- ISO 25.000 / ISO 9126:

### - Modelos de Referencia de Pruebas

- TRAMI: 5 modelos madurez para Test
- TPI: Modelo capacidad.

### TEMA 3: INDICADORES Y MÉTRICAS (Caja Blanca)

Diferencias de Caja Negra y Blanca: Ambas pueden estar codificadas en el código pero caja negra se centra en fuerza y caja blanca influye la calidad del código.

Una Métrica: Asignación de un valor a un atributo (tiempo, complejidad) de una entidad software.

Caja Blanca lo más olvidado siempre:

- No es un testing fácil, "si fueras no lo toques", no se ven resultados inmediatos, negativa a refactorizar.
- Efectos: Depender de solo 1 persona, para mantenibilidad coste en tiempo y dinero.

Testing de Regresión: Paso prueba y después toca el código, una y otra vez repetido. (Jenkins = Integración Continua)

Refactorizar → Test Unitarios → Tener Buon Diseño (Patrón State) 12

Métricas y Mediciones:

- Base: Las que no dependen de otra medida (líneas de código u horas programadas)
- Derivadas: Se sacan de una base (LOC por hora)

Prácticas Ágiles: Pair Review, Ping Pong Testing (1 hace otra revisa)

Mantener una rama de Git master → estable.

## Métricas Importantes:

- Número de Líneas de Código (LOC): 300 caso referenciado
- Complejidad Ciclomática: < 10 caso referenciado. Complejidad lógica de un programa.

Ramas Git: Mantener master estable y las ramas debes ser lo más cortas posibles

Problema: Presupuesto infinito y muchas clases → Cambiar las de más complejidad y a su vez las mas frecuencia de cambio / más dependientes al programa.

- ROI . Retorno de Inversión (Relación financiera con la refactorización)

Otras Métricas: Código-Repetido, Relación entre pruebas unitarias y complejidad Ciclomática

Tener varios niveles de abstracción (Facil tener mucha info de métricas, y luego no saber que hacer con ello)

## • TEMA 4: TESTING (Caja Negra)

Testing: Proceso que verifica la funcionalidad del proyecto y que garantiza calidad. (Que traga lo que se espera de él)

- Testing como tarea (Debe ser continuo) se considera una base en la visión arcaica (al final)
- Externalización del testing vs. Equipo multifuncional.

Importancia de la integración continua y control de versiones

3 Entornos: Desarrollo, Testing o preproducción y Producción.

- Continuous Delivery y continuos Deployment: Una versión lista siempre estable para poder desplegar
- DevOps: Romper la barrera que separa la producción y el test.

### Escuelas de Testing

1. Testing Formal: Muy matemático. Funciones y variables de entrada

2. Procesos y Estandares: Testing de Cascada (ISO)

3. Factoría (Externalización del Testing)

4. Escuela Context-Driven: Resalta el Humanismo.

- Humanos > Automatizar

- No todas las prácticas funcionan en todos los contextos

- Testing exploratorio: Ir probando

5. Testing Ágil: Persona vs. Proceso.

- Es una tarea / Equipo multifuncional.

- Ojo TDD, BDD, ATDD

- Diferencias entre testing y checking

- El testing es realizado por humanos (aunque se suele apoyar en herramientas)  
El checking lo puede hacer máquinas solas.

## TDD, BDD y ATDD

- TDD (Test Driven Development): Escribir primero la prueba que debe pasar el código y después la usa para simplificar el código.

No tiene porque ver con un H. Usuario

- BDD (Behavior Driven Development): Lo mismo pero con pruebas funcionales. (Siempre unido a Gherkin)

Proceso de dentro a afuera. BDD se puede hacer específico → By Example.

- ATDD: (Acceptable Test Driven Development): Todo el equipo discute los criterios de aceptación de la aplicación.

Asegura que es lo que quiere el usuario.

¿Una prueba unitaria es lo mismo que TDD? : No porque las pruebas unitarias se hacen al final y TDD no.