Unidad 5: Lineamiento de calidad

¿Qué significa calidad en software?

Un software de calidad no solo **funciona**, sino que además es:

- **Entendible**
- **Mantenible**
- **Escalable**
- **Confiable**

Analogía: Como una casa bien diseñada: no alcanza con que no se caiga, tiene que estar bien distribuida, ser segura y fácil de mantener.

Buenas prácticas de diseño de calidad

1. **Usar patrones de arquitectura conocidos**
2. **Representar claramente** datos, interfaces, componentes
3. **Modularidad**: dividir el sistema en partes lógicas
4. **Interfaces simples**: reducir la complejidad
5. **Notación clara y efectiva**

Modelo FURPS (de HP)

Clasifica atributos de calidad en dos grandes grupos:

Requisitos funcionales:

- **Funcionalidad (F)**: lo que hace el sistema, seguridad, interoperabilidad

Requisitos no funcionales:

- **Usabilidad (U)**: estética, documentación, facilidad de uso
- **Confiabilidad (R)**: robustez, disponibilidad, recuperación ante fallos
- **Rendimiento (P)**: eficiencia, velocidad, escalabilidad

- **Mantenibilidad (S)**: facilidad para adaptar, reparar y actualizar

Ventajas: fácil de entender y aplicar

Desventaja: requiere métricas, lo que implica más esfuerzo y costos

---

Conceptos Clave de Diseño (Pressman, Cap. 14)

- **Abstracción**: ocultar los detalles y mostrar lo esencial
- **Modularidad**: dividir el sistema en módulos funcionales
- **Ocultamiento de información**: cada módulo conoce lo necesario
- **Refinamiento**: evolución progresiva del diseño
- **Cohesión**: que un módulo haga **solo una cosa** bien
- **Acoplamiento**: que los módulos **dependan lo menos posible entre sí**

Control y Aseguramiento de la Calidad

🔷 Control de Calidad (QC)

- Verifica **si el producto** cumple con los requisitos
- Incluye pruebas y revisiones

🔷 Aseguramiento de Calidad (QA)

- Evalúa **si el proceso** de desarrollo es correcto
- Auditorías, checklist, revisiones sistemáticas

Dimensiones de Calidad (según Garvin)

Dimensión	Ejemplo / Significado
Desempeño	¿Cumple lo que promete?
Características	¿Ofrece algo extra o llamativo?
Confiabilidad	¿Es estable, libre de errores?
Conformidad	¿Sigue estándares?
Durabilidad	¿Es fácil de mantener a largo plazo?
Servicio	¿Es fácil de mantener y reparar?

Estética	¿Se ve y se siente bien?
Percepción	¿Cómo lo perciben los usuarios?

Técnicas de Revisión

Las revisiones detectan errores **tempranamente** (antes de codificar o entregar).

Tipos

- **Informales**: más rápidas, pero menos precisas
- **Formales (RTF)**: con roles definidos (moderador, escriba, lector, autor, inspectores)

Objetivos:

- Encontrar errores de lógica, implementación, funcionamiento
- Validar estándares
- Asegurar uniformidad del desarrollo

Métricas de revisión

- Horas de preparación, validación, repetición
- Tamaño del trabajo revisado (TPT)
- Densidad de errores: errores / TPT

Diferencia:

- **Error**: detectado antes de la entrega
- **Defecto**: detectado después de la entrega

Costo de la Calidad

Tipo de costo	Descripción
Prevención	Planificación, pruebas diseñadas
Evaluación	Pruebas ejecutadas, revisiones
Fallas internas	Costos antes de entregar el software

Fallas externas	Costos después de entregarlo (soporte, reputación)

Conclusión Final

> La calidad no es un accidente, sino el resultado de aplicar:

- Buenas prácticas de ingeniería
- Procesos bien definidos
- Técnicas de revisión efectivas
- Métricas claras y controladas

Producto de calidad = Buen diseño + Buen proceso + Entrega a tiempo + Usuario satisfecho