**📘 RESUMEN: Calidad en el Desarrollo de Software**

**✅ ¿Qué es la calidad en el desarrollo de software?**

La **calidad del software** se refiere a qué tan bien un producto de software cumple con lo que se espera de él. Es decir, si funciona correctamente, si es fácil de usar, si es seguro, y si puede adaptarse a cambios o ejecutarse en diferentes ambientes.

**🧩 Ejemplo simple:**

Si usas una app para pedir comida y esta:

* Te permite hacer el pedido (funcionalidad correcta),
* No se cae mientras la usas (fiabilidad),
* Es fácil de navegar (usabilidad),
* Y funciona tanto en Android como en iPhone (portabilidad),

… entonces estás frente a un software de buena calidad.

**🧱 ¿Qué es un modelo?**

Un **modelo** es como una guía que define cómo se debe trabajar para desarrollar un software de calidad. Es una **serie de documentos o prácticas organizadas** que ayudan a lograr buenos resultados.

**🧩 Ejemplo:**

Seguir una receta para cocinar es parecido a usar un modelo en desarrollo de software. Si la sigues paso a paso, el plato (software) saldrá bien.

**🧾 ¿Qué es un estándar?**

Un **estándar** es un conjunto de reglas o normas **oficialmente aprobadas por organizaciones internacionales** que te dicen cómo hacer bien un trabajo.

**🧩 Ejemplo:**

Así como existen reglas para construir un puente seguro, hay estándares (como ISO o IEEE) que ayudan a hacer software que funcione bien, sea seguro y fácil de mantener.

**🌍 Modelos más conocidos**

1. **ISO 9000**: Garantiza calidad en procesos generales, también aplicable al software.
2. **CMM (Capability Maturity Model)**: Evalúa qué tan maduros y organizados son los procesos de desarrollo.
3. **Bootstrap**: Modelo europeo que mezcla prácticas de CMM, ISO y SPICE.
4. **SPICE (ISO/IEC 15504)**: Estándar para mejorar y evaluar procesos de software.

**🔺 Modelo de McCall – Visión General**

El **modelo de McCall** fue uno de los primeros intentos de definir y medir la calidad del software. Se enfoca en **atributos que permiten evaluar si un software es bueno o no**.

Según este modelo, la calidad se divide en tres grandes bloques (ver **Figura 1**):

**1. Operación del producto**

Evalúa cómo se comporta el software cuando ya está en uso.

* **Corrección**: ¿Hace lo que debe hacer?
* **Fiabilidad**: ¿Lo hace bien todo el tiempo?
* **Eficiencia**: ¿Aprovecha bien los recursos?
* **Integridad**: ¿Es seguro?
* **Facilidad de uso**: ¿Es fácil de manejar?

**2. Revisión del producto**

Mide qué tan fácil es modificar, probar o mantener el software.

* **Mantenibilidad**: ¿Es fácil de arreglar?
* **Prueba**: ¿Se puede probar fácilmente?
* **Flexibilidad**: ¿Se puede adaptar a cambios?

**3. Transición del producto**

Evalúa si el software puede moverse a otros entornos o ser reutilizado.

* **Interoperabilidad**: ¿Puede comunicarse con otros sistemas?
* **Portabilidad**: ¿Funciona en otros equipos o plataformas?
* **Reusabilidad**: ¿Se pueden usar partes en otro software?



**🧠 Factores de calidad según McCall – Relación con el cliente, la dirección y el desarrollador**

En la **Figura 2**, se muestra cómo los diferentes factores de calidad son vistos desde tres perspectivas:

**1. Visión del cliente**

El cliente busca:

* Que el software **funcione bien** (corrección).
* Que sea **seguro** (integridad).
* Que **sea fácil de usar** y no falle (usabilidad, fiabilidad).

**2. Visión de la dirección**

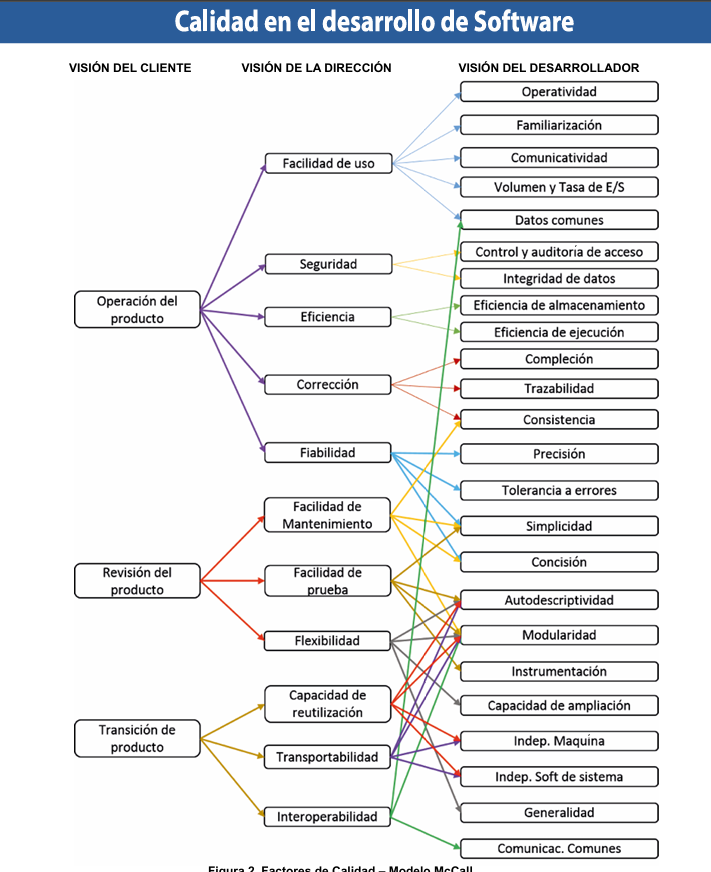
A la gerencia le interesa:

* Que el software sea **eficiente y mantenible** (bajo costo, poco tiempo de arreglo).
* Que **pueda crecer o cambiar** sin rehacer todo.

**3. Visión del desarrollador**

El desarrollador se enfoca en:

* **Modularidad** (partes independientes).
* **Trazabilidad** (seguir los cambios).
* **Tolerancia a errores** (que no se dañe si algo sale mal).
* **Precisión, consistencia y simplicidad** en el código.



**🎯 Conclusión**

El modelo de McCall es una herramienta muy útil para **entender y evaluar la calidad del software** desde distintas perspectivas: cliente, gerencia y desarrollo.

Aplicando modelos y estándares como McCall, ISO o CMMI, se puede garantizar que el software que se construye no solo **funcione bien**, sino que también sea **seguro, eficiente, reutilizable y fácil de mantener**.