# 📘 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**

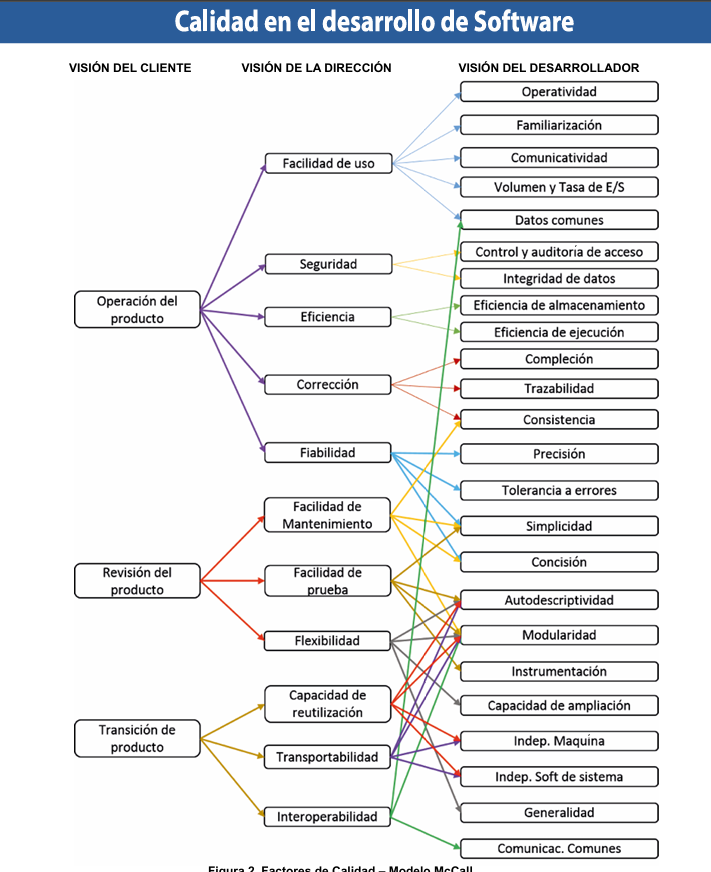
A la gerencia le interesa:

* 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**.

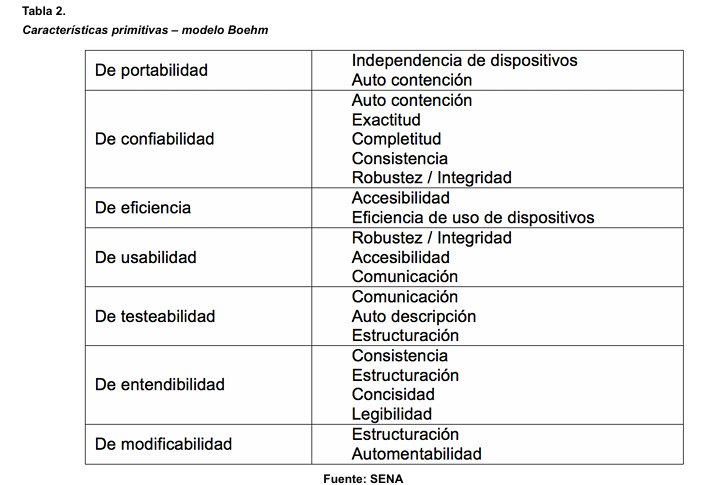
## 🧠 Modelo Boehm

### ✅ ¿Qué es el modelo Boehm?

El **modelo Boehm** es un marco que ayuda a evaluar la calidad del software en tres niveles:

1. Nivel Alto
2. Nivel Intermedio
3. Nivel Primitivo

Es útil tanto para los usuarios (porque analiza si el software sirve y es fácil de usar), como para los desarrolladores (porque revisa si es fácil de entender, probar y mantener).



Diagrama

El contenido generado por IA puede ser incorrecto.

### 📌 Niveles del modelo Boehm

#### 1. Nivel Alto: Utilidad General

Es la calidad global del software, es decir, ¿es útil realmente?

📍*Ver: Figura 3 (Modelo Boehm)*

#### 2. Nivel Intermedio: Se divide en dos partes:

* **Utilidad Percibida:** lo que el usuario nota directamente.
  + Ejemplo: Si una aplicación bancaria es confiable, rápida y se puede usar en diferentes dispositivos.
* **Facilidad de Mantenimiento:** lo que facilita el trabajo de los programadores y técnicos.
  + Ejemplo: Si se puede modificar fácilmente un módulo del software sin dañar los otros.

📍*Ver: Figura 3*

#### 3. Nivel Primitivo: Detalles técnicos

Aquí se descomponen las características en propiedades específicas que permiten medir la calidad con más precisión.

📍*Ver: Tabla 2*

**🧩 Explicación de cada característica con ejemplos**

#### 🔷 Portabilidad

* ¿Funciona el software en diferentes dispositivos o sistemas?
* 📌 Ej: Una app que corre igual en Android, iOS y PC.
* Primitivas: *Independencia de dispositivos, Auto contención*

#### 🔷 Confiabilidad

* ¿Hace el software lo que se espera sin fallos?
* 📌 Ej: Un sistema de nómina que calcula los sueldos siempre correctamente.
* Primitivas: *Exactitud, Completitud, Consistencia, Robustez/Integridad*

#### 🔷 Eficiencia

* ¿Utiliza bien los recursos del sistema? ¿Es rápido?
* 📌 Ej: Una app que no consume mucha batería ni se pone lenta.
* Primitivas: *Accesibilidad, Eficiencia de uso de dispositivos*

#### 🔷 Usabilidad (Ingeniería Humana)

* ¿Es fácil de aprender y usar?
* 📌 Ej: Que el botón de “guardar” sea fácil de encontrar.
* Primitivas: *Accesibilidad, Comunicación, Robustez/Integridad*

#### 🔷 Testeabilidad (Facilidad de prueba)

* ¿Es fácil de probar para encontrar errores?
* 📌 Ej: Un sistema con módulos bien separados facilita las pruebas automáticas.
* Primitivas: *Comunicación, Auto descripción, Estructuración*

#### 🔷 Entendibilidad (Facilidad de comprensión)

* ¿Es fácil de leer y entender el código y la lógica del sistema?
* 📌 Ej: Un código con nombres claros y bien comentado.
* Primitivas: *Concisión, Legibilidad, Estructuración, Consistencia*

**🔷 Modificabilidad (Facilidad de cambio)**

* ¿Es fácil actualizar o modificar el software?
* 📌 Ej: Cambiar el diseño de un formulario sin afectar otras partes del sistema.
* Primitivas: *Estructuración, Automentabilidad*

## 📌 Conclusión

El modelo Boehm permite evaluar la calidad del software en profundidad.  
🔹 **Los usuarios** obtienen software útil, confiable y fácil de usar.  
🔹 **Los desarrolladores** logran sistemas que pueden mantener, probar y modificar sin complicaciones.

## ¿Qué es CMMI?

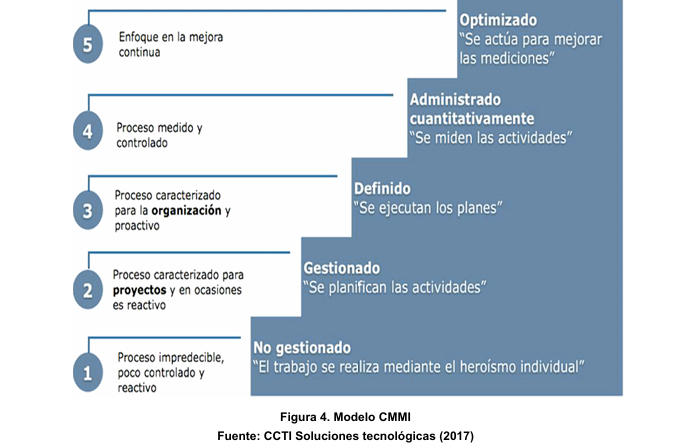
**CMMI (Capability Maturity Model Integration)** es un **modelo de mejora de procesos** que ayuda a las organizaciones (especialmente las que desarrollan software) a optimizar su rendimiento, mediante buenas prácticas organizadas en niveles de madurez.

Su propósito es proporcionar una **guía clara para gestionar, desarrollar y mantener procesos eficientes**, con el fin de mejorar la calidad, reducir riesgos y aumentar la productividad.

### **🎯** Objetivos principales del CMMI

* **Trazar cinco niveles de madurez** que permitan evaluar la capacidad de los procesos en una organización.
* **Guiar a las organizaciones** en el control y mejora de sus procesos de desarrollo y mantenimiento.
* **Identificar prácticas clave** que ayuden en la mejora continua.
* **Diagnosticar la madurez actual** de una organización y señalar los elementos críticos para mejorar.
* Aunque CMMI tiene una visión amplia, su base original (CMM) se enfocaba principalmente en el **modelo en cascada** del desarrollo de software.

### 🪜 Niveles de Madurez en el Modelo CMMI

🖼️ **Ubicación de la imagen:**  


📌 *Figura 1. Modelo CMMI – Fuente: CCTI Soluciones Tecnológicas (2017)*

#### 🔵 Nivel 1 – Inicial (No gestionado)

* **Descripción:**  
  Procesos caóticos, impredecibles y mal controlados.
* **Frase clave:** *“El trabajo se realiza mediante el heroísmo individual.”*
* **Ejemplo:**  
  Un equipo de desarrollo empieza a crear software sin planificación formal. Cada uno trabaja a su manera, y los resultados son inconsistentes. Si el líder del equipo se ausenta, todo se paraliza.

#### 🟢 Nivel 2 – Gestionado

* **Descripción:**  
  Hay consciencia de la dirección. Los procesos se gestionan, aunque de forma básica y por proyecto.
* **Frase clave:** *“Se planifican las actividades.”*
* **Ejemplo:**  
  Una empresa empieza a documentar tareas, asignar responsables y hacer seguimiento con cronogramas. Cada proyecto tiene su propia manera de hacerlo, pero al menos hay control.

#### 🟡 Nivel 3 – Definido

* **Descripción:**  
  Los procesos están estandarizados, documentados y se comprenden a nivel organizacional.
* **Frase clave:** *“Se ejecutan los planes.”*
* **Ejemplo:**  
  Todos los equipos de la empresa usan la misma metodología para desarrollar software (como Scrum o RUP), con manuales, plantillas y prácticas comunes.

#### 🟠 Nivel 4 – Administrado Cuantitativamente (Gestión Cuantitativa)

* **Descripción:**  
  La organización establece **objetivos medibles** para controlar el rendimiento y calidad de los procesos.
* **Frase clave:** *“Se miden las actividades.”*
* **Ejemplo:**  
  Se recopilan métricas como defectos por módulo, tiempo promedio de respuesta, velocidad de desarrollo, etc. Estas métricas ayudan a tomar decisiones basadas en datos.

#### 🔴 Nivel 5 – Optimizado

* **Descripción:**  
  Se trabaja en la **mejora continua de los procesos**, aplicando innovación y acciones correctivas basadas en los datos recogidos.
* **Frase clave:** *“Se actúa para mejorar las mediciones.”*
* **Ejemplo:**  
  La empresa usa inteligencia artificial o aprendizaje automático para predecir retrasos o errores, automatiza pruebas y mejora sus procesos continuamente, según análisis de tendencias.

### 📘 Ejemplo resumido tipo historia:

Una pequeña empresa comienza su camino en CMMI...

1. Al principio (Nivel 1), cada desarrollador trabaja por su cuenta, y los proyectos son un caos.
2. Luego (Nivel 2), empiezan a usar tareas con responsables y cronogramas.
3. Más adelante (Nivel 3), adoptan una metodología común y entrenan a todos.
4. Con el tiempo (Nivel 4), implementan métricas de calidad y rendimiento.
5. Finalmente (Nivel 5), mejoran continuamente con base en análisis y tecnología.

### ✅ ¿Por qué es útil CMMI?

* Mejora la calidad de los productos y servicios.
* Reduce riesgos de fallos y sobrecostos.
* Permite certificaciones que demuestran madurez ante clientes.
* Fomenta la cultura de mejora continua.

# 📘Calidad en el Desarrollo de Software (ISO/IEC 9126)

## 📌 ¿Qué es el estándar ISO/IEC 9126?

Es una guía creada para evaluar la **calidad del software**. Fue desarrollada en 2001 y actualizada por última vez en 2004.

Evalúa la calidad **desde seis grandes factores o atributos**, que se dividen en subcaracterísticas. Estos atributos pueden ser:

* **Externos:** Se notan cuando el software ya se está usando (por los usuarios).
* **Internos:** Se revisan antes de lanzar el software, evaluando su estructura y comportamiento.

### ✅ Factores de Calidad del Software y sus Subcaracterísticas

A continuación te explico cada uno de los factores con ejemplos sencillos:

#### 1. Funcionalidad

👉 ¿El software hace lo que debe hacer?  
Subcaracterísticas:

* **Adecuación:** ¿El software tiene las funciones necesarias?  
  *Ejemplo:* Un cajero automático debe permitir retirar dinero, consultar saldo y transferir.
* **Exactitud:** ¿Los resultados que da el software son correctos?  
  *Ejemplo:* Si haces una suma en una app de contabilidad, ¿el resultado es preciso?
* **Seguridad:** ¿Protege bien los datos y evita accesos no autorizados?  
  *Ejemplo:* Un banco online que pide verificación por mensaje para iniciar sesión.
* **Interoperabilidad:** ¿Se puede comunicar con otros sistemas?  
  *Ejemplo:* Un sistema de ventas que se conecta con otro sistema para facturar.
* **Cumplimiento:** ¿Cumple con normas legales o reglas del sector?  
  *Ejemplo:* Una tienda virtual debe emitir factura legal según la ley local.

#### 2. Confiabilidad

👉 ¿Qué tan estable es el software?  
Subcaracterísticas:

* **Madurez:** ¿Se cae o falla seguido?  
  *Ejemplo:* Una app que se cierra sola constantemente tiene baja madurez.
* **Tolerancia a fallas:** ¿Sigue funcionando bien si ocurre un error?  
  *Ejemplo:* Una calculadora que avisa si el número ingresado no es válido en vez de cerrarse.
* **Recuperación:** ¿Puede volver a funcionar luego de un error?  
  *Ejemplo:* Word guarda versiones automáticas por si se cierra inesperadamente.
* **Cumplimiento:** Igual que en funcionalidad, debe cumplir normas relacionadas a la confiabilidad.

#### 3. Eficiencia

👉 ¿Qué tan bien usa los recursos y qué tan rápido trabaja?  
Subcaracterísticas:

* **En tiempo:** ¿Responde rápido?  
  *Ejemplo:* Un motor de búsqueda que muestra los resultados en segundos.
* **En recursos:** ¿Usa poca memoria o batería?  
  *Ejemplo:* Una app que no ralentiza tu celular ni lo calienta mucho.

#### 4. Usabilidad

👉 ¿Es fácil de usar?  
Subcaracterísticas:

* No aparecen en esta parte del cuadro, pero en el estándar se incluyen cosas como:
  + Facilidad de aprendizaje
  + Facilidad para operar
  + Ayuda y mensajes claros
  + Diseño amigable

*Ejemplo:* Una app como WhatsApp que cualquiera puede entender sin leer un manual.

#### 5. Mantenibilidad

👉 ¿Qué tan fácil es corregirlo o actualizarlo?  
Subcaracterísticas:

* No aparece aquí, pero implica:
  + Facilidad de modificar
  + Facilidad para detectar errores
  + Facilidad para mejorar el software

*Ejemplo:* Un sistema donde cambiar un menú o corregir una función no toma mucho tiempo.

#### 6. Portabilidad

👉 ¿Se puede usar en otros dispositivos o sistemas?  
Subcaracterísticas:

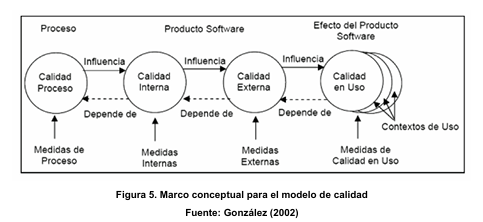
* Tampoco aparece aquí, pero incluye:
  + Adaptabilidad (funciona en distintos equipos)
  + Instalabilidad
  + Reemplazabilidad

*Ejemplo:* Una app que funciona en Android, iPhone y tablets.

## 📝 Conclusión

El estándar **ISO/IEC 9126** te ayuda a entender qué hace que un software sea de **buena calidad**. Así no solo se mide si “funciona”, sino **cómo funciona**, **qué tan seguro es**, **si es rápido**, **si se puede confiar en él**, etc.

📍 **Consejo para estudiar:** Intenta ver cada uno de estos factores en los programas o apps que usas día a día (por ejemplo: Gmail, Instagram, Excel). Así relacionas la teoría con la vida real.



**1. Calidad del Proceso**

* 🔹 Se refiere a **cómo se hace el software**: el proceso de desarrollo, pruebas, control de errores, etc.
* 🔸 **Importancia**: Un buen proceso aumenta las probabilidades de que el software final sea de calidad.
* 🧪 Se mide con: **Medidas del proceso**
  + *Ejemplo:* ¿Se siguen buenas prácticas de programación? ¿Se revisa el código antes de lanzarlo?

**2. Calidad Interna**

* 🔹 Es la calidad que **se puede evaluar desde dentro del software**, sin necesidad de usarlo.
* 🔸 Se refiere a aspectos técnicos como: estructura del código, claridad, organización, modularidad.
* 🧪 Se mide con: **Medidas internas**
  + *Ejemplo:* Un software con código limpio, bien documentado y sin errores de compilación.

**3. Calidad Externa**

* 🔹 Es la calidad que **se observa cuando el software ya está funcionando**, pero desde afuera (por los usuarios o sistemas que interactúan con él).
* 🔸 Depende de la calidad interna, pero ahora se ve cómo responde el software.
* 🧪 Se mide con: **Medidas externas**
  + *Ejemplo:* ¿Responde rápido? ¿Se conecta bien con otros sistemas? ¿Es seguro?

**4. Calidad en Uso**

* 🔹 Es el **impacto real del software** en la vida del usuario.
* 🔸 Aquí se mide si el software realmente ayuda a cumplir una tarea, mejora la productividad, evita errores humanos, etc.
* 🧪 Se mide con: **Medidas de calidad en uso**, y depende del **contexto** en el que se utilice (por ejemplo: empresa, escuela, industria, etc.)
  + *Ejemplo:* Una app médica debe ser muy precisa y segura para no cometer errores con los pacientes.

**🔁 ¿Cómo se relacionan?**

* 🔄 **Cada etapa depende de la anterior**:  
  Si el proceso es malo → probablemente la calidad interna será mala → lo que llevará a una mala calidad externa → y finalmente, el usuario no estará satisfecho.

**📌 Conclusión sencilla:**

"Si queremos que un software sea útil y funcione bien para el usuario final, necesitamos empezar con un buen proceso de desarrollo. Cada paso afecta el siguiente."

# 🧸 ¿Qué es el estándar ISO/IEC 25000?

Imagina que estás haciendo un **juguete** (como un robot). Para que ese robot funcione bien y no se rompa, necesitas seguir ciertas reglas, usar buenas piezas y revisar que todo esté perfecto antes de dárselo a alguien para jugar.

💻 En el mundo del software pasa igual:  
El estándar ISO/IEC 25000 es **un conjunto de reglas** que nos ayudan a asegurarnos de que los programas (como las apps o videojuegos) sean:

* Útiles,
* Fáciles de usar,
* Rápidos,
* Y que no se dañen fácilmente.

**🧩** ¿De qué se compone?

La familia ISO 25000 tiene **cinco partes**, como si fueran piezas de un rompecabezas:

### **1.** 🧑‍🏫 ISO 2500n – Gestión de la calidad

👉 Ayuda a planear y organizar cómo se va a hacer el software de buena calidad.

**Ejemplo:** Como cuando mamá hace una lista para que el pastel salga rico (harina, huevos, horno, etc.).

### 2. 🧱 ISO 2501n – Modelo de calidad

👉 Dice **qué cosas hacen que un software sea de buena calidad**.

**Ejemplo:** Si hiciste un juguete, ¿funciona? ¿Es seguro? ¿Es bonito?  
En software, sería: ¿funciona bien? ¿Es rápido? ¿Es fácil de usar?

### 3. 🔍 ISO 2502n – Medida de la calidad

👉 Enseña cómo **medir** si el software está bien hecho o no.

**Ejemplo:** Como medir con una regla si un bloque tiene el tamaño correcto.  
Aquí, se miden cosas como: tiempo de respuesta, cantidad de errores, etc.

### 4. 📄 ISO 2503n – Requisitos de calidad

👉 Explica **qué cosas necesita tener el software** para ser útil y bueno.

**Ejemplo:** Si haces una pista de carros, debes poner rampas, curvas y señales.  
En software, estos son los requisitos: qué debe hacer y cómo.

### 5. 🎯 ISO 2504n – Evaluación de la calidad

👉 Ayuda a **revisar y comprobar** que todo esté bien antes de entregar el software.

**Ejemplo:** Como probar un juguete antes de regalarlo. ¿Funciona? ¿Está completo?

## 🧠 ¿Por qué es importante?

Porque ayuda a que **todos los programas y sistemas que usamos (apps, páginas, juegos)**:

* No tengan errores,
* Sean seguros,
* Y funcionen como deben.

## 📌 En resumen:

| **Parte ISO** | **¿Qué hace?** | **Ejemplo sencillo** |
| --- | --- | --- |
| 2500n | Organiza la calidad | Lista de compras del pastel |
| 2501n | Dice qué es “bueno” | ¿El juguete funciona bien? |
| 2502n | Mide la calidad | Usar regla o cronómetro |
| 2503n | Qué debe tener el software | ¿Qué piezas lleva el robot? |
| 2504n | Revisa si todo está bien | ¿El juguete se puede usar? |

# 🧠 ¿Qué es el ISO/IEC 25010?

Es una **actualización del estándar anterior ISO/IEC 9126**, y sirve para decir **qué cosas debe tener un software para considerarse de buena calidad**.

## 🎯 ¿Qué cosas evalúa este estándar?

Evalúa **dos grandes modelos** de calidad:

### 1. 📱 Modelo de Calidad en Uso

Este modelo **mide cómo se siente y funciona el software cuando alguien lo usa**.  
👉 Se enfoca en la experiencia del usuario final.

📌 Tiene **5 características** principales (algunas con subcaracterísticas).

**Ejemplo sencillo:**  
Usas una app para pedir comida:

* ¿Es fácil de usar?
* ¿Te ayudó a resolver tu necesidad?
* ¿Te evitó errores?
* ¿Te sientes seguro al usarla?

### 2. 🖥️ Modelo de Calidad del Producto

Este modelo **mide lo que tiene el software por dentro y por fuera**.  
👉 Evalúa las partes internas del sistema (cómo está hecho) y las externas (cómo se comporta).

📌 Tiene **8 características** que se dividen en **31 subcaracterísticas**.

**Ejemplo sencillo:**  
Piensa en un robot que armamos:

* ¿Funciona bien?
* ¿Está bien ensamblado?
* ¿Es seguro?
* ¿Es fácil de actualizar?
* ¿Se puede usar en otro país?

### **💡 ¿Por qué son importantes estas características?**

Porque permiten:

* Medir si el software cumple lo que promete.
* Comparar programas entre sí.
* Hacer mejoras continuas.
* Tener un lenguaje común para hablar de calidad.

## 📦 En resumen:

| **Modelo** | **¿Qué evalúa?** | **Ejemplo** |
| --- | --- | --- |
| **Calidad en uso** | La experiencia al usar el software | ¿Pudo el usuario hacer su pedido fácil? |
| **Calidad del producto** | Cómo está hecho y cómo se comporta | ¿El robot está bien armado y responde bien? |



## 🧩 Datos clave del estándar ISO/IEC 25010:2011:

* Sustituye al viejo estándar ISO 9126.
* Da claridad en cómo **medir, evaluar y mejorar la calidad**.
* Ayuda a tener un lenguaje claro y común entre desarrolladores, empresas y usuarios.

# 📘 ¿Qué es el estándar ISO/IEC 15504 – SPICE?

🔧 **SPICE** (Software Process Improvement and Capability dEtermination) es un **modelo para mejorar y evaluar cómo se desarrollan y mantienen los programas o sistemas de software**.

📍 Se usa mucho en **Europa** y está basado en buenas prácticas como las de **CMMI**.

## 🔎 ¿Para qué sirve SPICE?

SPICE **no te dice exactamente cómo hacer las cosas**, pero **te da una guía para evaluar y mejorar tus procesos**. Sirve para:

1. ✅ **Mejorar procesos**: ver qué se puede hacer mejor al crear software.
2. 🧪 **Evaluar procesos**: revisar cómo se está trabajando en un proyecto.
3. 📊 **Medir la capacidad**: saber qué tan bien una empresa puede desarrollar software de forma repetida y controlada.

🧠 También **se alinea con ISO 12207**, que define todo el **ciclo de vida del software** (desde su planeación hasta su mantenimiento).

## 📦 ¿Qué contiene SPICE?

SPICE está formado por **nueve partes (documentos)** que **explican todo el modelo** y **cómo evaluarlo**. La imagen que enviaste los muestra gráficamente. Te los explico de forma fácil:

| **Parte** | **Contenido** | **Explicación Sencilla** |
| --- | --- | --- |
| **Parte 1** | Conceptos y guía introductoria | Qué es SPICE y para qué sirve. |
| **Parte 2** | Modelo para administración de procesos | Qué procesos debes tener y cómo se organizan. |
| **Parte 3** | Proceso de calificación | Cómo se mide y clasifica la calidad de los procesos. |
| **Parte 4** | Guía para la evaluación | Pasos para hacer una evaluación formal. |
| **Parte 5** | Selección de herramientas | Qué herramientas usar para medir y evaluar. |
| **Parte 6** | Calificación y entrenamiento de asesores | Cómo formar a los que evalúan procesos. |
| **Parte 7** | Uso para mejorar procesos | Cómo aplicar SPICE para mejorar lo que ya haces. |
| **Parte 8** | Evaluar proveedores | Cómo saber si una empresa externa trabaja bien. |
| **Parte 9** | Vocabulario | Significados claros de todos los términos usados. |

## 🎯 ¿Para quién es útil SPICE?

* Para empresas que desarrollan software.
* Para quienes evalúan la calidad del trabajo en TI.
* Para proveedores que quieren demostrar que trabajan bien.
* Para mejorar continuamente cómo se hacen los proyectos.

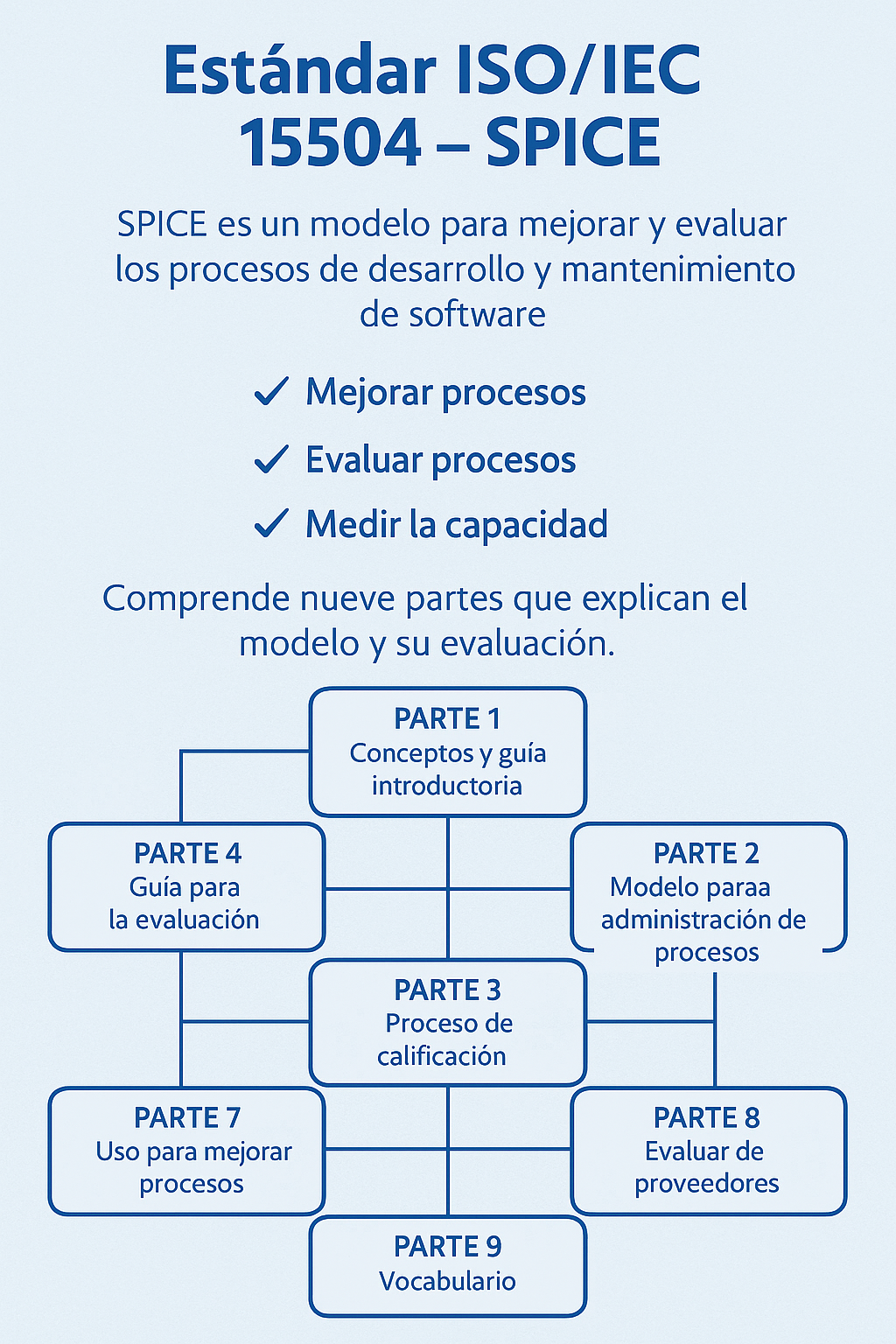
## 🧠 Ejemplo práctico

Imagina que tienes una empresa que hace apps. Usando SPICE:

* Evalúas si tus programadores siguen buenas prácticas.
* Mides si los errores bajan con cada entrega.
* Revisas si tus procesos están bien definidos.
* Usas esa información para mejorar poco a poco.

## 📝 En resumen:

* SPICE es una guía para **evaluar y mejorar cómo se hace software**.
* No es una receta exacta, sino una **estructura para medir, comparar y crecer**.
* Ayuda a lograr productos de mejor calidad de manera **más ordenada y profesional**.



# 📘 ¿Qué es ISO 9001:2015?

ISO 9001:2015 es un estándar internacional que ayuda a las organizaciones a asegurarse de que sus productos y servicios **cumplen con las necesidades del cliente** y **mejoran continuamente**.

🧠 Se enfoca en la **gestión de calidad**: hacer bien las cosas desde el principio, cumpliendo con lo importante, no solo con lo obligatorio.

## 📊 ¿Qué cambió del ISO 9001:2008 al ISO 9001:2015?

| **ISO 9001:2008 (Antes)** | **ISO 9001:2015 (Ahora)** |
| --- | --- |
| ✅ Cumplimiento de requisitos legales y normas | ✅ Enfoque en lo importante para el negocio, sus clientes y el entorno |
| 📚 Formación y documentación | 💬 Liderazgo, conciencia, comunicación y habilidades. Se trata de **"querer hacerlo"** |
| 🔧 Acciones preventivas | ⚠️ Gestión de riesgos. Detectar problemas antes de que ocurran |
| 📄 Manuales, procedimientos y registros | 📂 Información documentada pero más flexible (lo importante, no el papeleo) |
| 🛒 Control de compras | 🔄 Control de bienes y servicios externos que afectan la calidad del producto final |

# 🧩 Explicado de forma fácil con ejemplos:

| **Aspecto** | **Ejemplo práctico en una empresa de software** |
| --- | --- |
| **Enfoque en negocio y contexto** | Se revisa si el software que haces realmente resuelve las necesidades del cliente. |
| **Liderazgo y cultura de calidad** | Los líderes motivan al equipo y fomentan buena comunicación, no solo imponen reglas. |
| **Gestión de riesgos** | Antes de lanzar una app, identificas posibles fallos en dispositivos viejos y los previenes. |
| **Documentación más útil** | Guardas solo los documentos que realmente ayudan al equipo, no se guarda "por cumplir". |
| **Relación con proveedores** | Si tu software depende de una API externa, verificas que esa API cumpla con lo esperado. |

# 📌 En resumen:

* **ISO 9001:2015** es más **moderno y flexible**.
* Ya no se trata solo de **seguir reglas**, sino de **comprender el negocio, mejorar continuamente y evitar riesgos**.
* Se pone énfasis en **liderazgo, contexto, objetivos estratégicos y valor real para el cliente**.

# Aseguramiento de la calidad de software.

## 🧩 1. ¿Qué es el Aseguramiento de la Calidad del Software (SQA)?

El SQA es un conjunto de actividades sistemáticas y planificadas que aseguran que el software cumpla con los estándares de calidad y satisfaga los requisitos definidos.

🔧 **Incluye:**

* Gestión de calidad.
* Herramientas y métodos de ingeniería.
* Revisiones técnicas.
* Pruebas.
* Control de documentación.
* Cumplimiento de estándares.
* Informes y métricas.

💡 **Ejemplo:** En una empresa fintech, el equipo SQA revisa cada cambio en el sistema bancario para asegurar que cumpla con normas de seguridad y confiabilidad.

## 🎯 2. Objetivos del SQA

1. Planificar las actividades.
2. Verificar cumplimiento de estándares.
3. Informar a los equipos.
4. Comunicar desviaciones no resueltas.

💡 **Ejemplo:** Si un módulo no cumple con los requisitos técnicos, el SQA lo detecta y reporta antes de que llegue al cliente.

## 🧑‍🤝‍🧑 3. Roles en el grupo SQA

* **Líder de proyecto:** Coordina el equipo.
* **Administrador de desarrollo:** Supervisa diseño y codificación.
* **Administrador de planificación:** Establece y verifica planes.
* **Administrador de apoyo:** Proporciona herramientas.
* **Administrador de calidad:** Define el plan de calidad.
* **Desarrolladores:** Ejecutan tareas técnicas.

💡 **Ejemplo:** El administrador de calidad crea un plan para validar cada nueva funcionalidad con pruebas y auditorías.

## 📋 4. Actividades del grupo SQA

* Crear el plan SQA.
* Revisar procesos de ingeniería.
* Auditar productos.
* Documentar desviaciones.
* Medir métricas.
* Control de configuración.

💡 **Ejemplo:** En una app móvil, el grupo SQA documenta cada error detectado y verifica si ya fue solucionado antes de publicar la app.

## 📑 5. Plan de SQA

Documento que describe cómo asegurar la calidad del software, incluyendo auditorías, seguimiento y cumplimiento.

💡 **Ejemplo:** Un plan SQA para una plataforma e-commerce incluye pruebas de seguridad, carga y revisión de código.

## 🔍 6. Revisión de software

Se realiza luego del diseño/codificación para:

* Detectar errores.
* Establecer qué mejorar.
* Obtener calidad uniforme.
* Servir como modelo para detectar fallos.

💡 **Ejemplo:** Después de crear una calculadora web, un revisor detecta que ciertos resultados no se redondean correctamente y lo reporta.

## 📊 7. Métricas

Las métricas ayudan a evaluar, mejorar y predecir resultados en el proceso de desarrollo.

💡 **Ejemplo:** Se mide la cantidad de errores por cada 1000 líneas de código para evaluar la calidad del producto.

## 🔄 8. Fases del ciclo de vida para SQA

| **Fase** | **Actividad del grupo SQA** | **Ejemplo** |
| --- | --- | --- |
| Planificación | Crear plan SQA y auditorías | Verificar que se contemplen pruebas de carga |
| Requerimientos | Validar claridad y completitud | Confirmar que todos los requisitos del usuario estén documentados |
| Diseño y desarrollo | Asegurar estándares y módulos | Validar que cada módulo del sistema esté incluido |
| Implementación | Auditar entregables y configuración | Verificar que la versión publicada esté actualizada |
| Integración y prueba | Validar resultados de pruebas | Comprobar que pruebas unitarias y de sistema se completaron |
| Aceptación y entrega | Auditar configuración final | Garantizar que el software esté listo para el cliente |

## 📚 9. Modelos y estándares

**🧭 CMM - CMMI**

Modelo de madurez con 5 niveles. Evalúa procesos y mejora la organización.

💡 **Ejemplo:** Una empresa en nivel 2 ya documenta procesos, mientras que una en nivel 5 los optimiza continuamente.

**🏅 ISO**

* **ISO 9001:** Enfocada en la satisfacción del cliente y procesos organizacionales.
* **ISO 9000-3:** Calidad en todas las fases del software.
* **ISO 9004-2:** Mejora continua de desempeño.
* **ISO/IEC 12207:** Define procesos del ciclo de vida del software.
* **ISO/IEC 15504 (SPICE):** Evalúa y mejora procesos, común en Europa.

💡 **Ejemplo:** Una empresa certificada ISO 9001 demuestra que puede entregar software consistente con las expectativas del cliente.

**🔌 IEEE**

| **Norma** | **Propósito** |
| --- | --- |
| IEEE 730 | Establece contenidos mínimos del plan SQA |
| IEEE 829 | Define documentación del proceso de pruebas |
| IEEE 830 | Guía para redactar especificaciones |
| IEEE 1012 | Verificación y validación |
| IEEE 1061 | Métricas de calidad de software |

💡 **Ejemplo:** Para un nuevo módulo de banca digital, se usa IEEE 829 para documentar pruebas funcionales.