TÉCNICAS DE CONTROL DE CALIDAD DEL SOFTWARE

1. Control de Calidad del Software

La calidad del software se define como el grado en que un producto cumple con requisitos explícitos e implícitos del usuario, considerando atributos como fiabilidad, usabilidad y mantenibilidad. Incluye dos perspectivas:

- Funcionalidad técnica: Cumplimiento de especificaciones técnicas (ejemplo: procesar 1,000 transacciones/segundo).
- Experiencia de usuario: Ausencia de fallas percibidas (ejemplo: mensajes de error comprensibles).

Un error común es suponer que "funcionar técnicamente" equivale a "no fallar". Por ejemplo, un mensaje de error genérico como "Error xx00180" no resuelve la frustración del usuario, mientras que una guía clara ("Reinicie la aplicación y actualice el sistema") preserva la percepción de calidad.

Calidad del Software: qué es, qué la compone y estándares principales.

La calidad del software es un pilar fundamental en el desarrollo de productos tecnológicos que no solo cumplan con las expectativas del cliente, sino que también sean fiables y eficientes. En este material, explicaremos qué es la calidad del software, los elementos clave que la componen y los principales estándares que guían su implementación en las organizaciones.

Historia de la Calidad del Software

El impulso por la calidad en el desarrollo de software comenzó en la década de 1980 debido a la rápida evolución tecnológica y la creciente dependencia de sistemas críticos.

Un evento clave fue el fallo de un misil británico en la Guerra de las Malvinas en 1982, causado por un error de software, lo que subrayó la necesidad de métodos estrictos para garantizar la calidad en sistemas críticos.

Además, la administración de Margaret Thatcher en el Reino Unido implementó políticas que exigían estándares de alta calidad en productos de TIC, impulsando la creación de normas internacionales para asegurar la fiabilidad del software.

Ambos eventos resaltaron la urgencia de desarrollar métodos estandarizados para garantizar la calidad en el desarrollo de software.

Para resolver estas iniciativas diversas instituciones trabajaron en la creación y aplicación de estándares de gran importancia para el desarrollo de software como:

1. CMMI (*Capability Maturity Model Integration*): Desarrollado por el *Software Engineering Institute* (SEI) de la Universidad Carnegie Mellon, este modelo proporciona un marco para mejorar la capacidad de los procesos en la gestión y desarrollo de software, con el objetivo de alcanzar altos niveles de madurez y calidad en las organizaciones.
2. ISO (*International Organization for Standardization*): Esta organización internacional desarrolla y publica estándares que aseguran la calidad y seguridad en diversas industrias. Algunas de las normas aplicables al desarrollo de software son:
    1. ISO 9000: Define principios básicos para sistemas de gestión de calidad, aunque es una norma “más genérica”, es aplicable en cualquier industria, incluyendo software.
    2. ISO 12207: Establece procesos para el ciclo de vida del software, desde el diseño hasta el mantenimiento.
    3. ISO 10005: Guía para crear planes de calidad en proyectos, asegurando el cumplimiento de requisitos.
    4. ISO 29110: Marco simplificado para el desarrollo de software en pequeñas organizaciones.
    5. ISO/IEC/IEEE 29119: Proporciona un enfoque estandarizado para la prueba de software.
3. IEEE (Institute of Electrical and Electronics Engineers): Reconocido a nivel mundial, el IEEE establece estándares técnicos para una amplia gama de tecnologías, incluyendo software. Sus estándares, como el ya mencionado ISO/IEC/IEE 29119 y el IEEE 730 buscan asegurar que los productos tecnológicos y de software sean seguros, confiables y de alta calidad, lo que contribuye a la mejora continua en la ingeniería de software.

¿Qué es calidad del software?

La calidad, en términos generales, se refiere al grado en que un producto o servicio **cumple con los requisitos y expectativas establecidos por el cliente o el usuario final**. Implica la conformidad con especificaciones y la capacidad de satisfacer necesidades de manera consistente y confiable.

Cómo Realizar una Buena Especificación de Requisitos de Software

La claridad en la definición y comprensión de lo que tus clientes esperan es la base de la calidad de tus soluciones de software. Especificar y validar estos requisitos permitirá a tu equipo ser más creativo y productivo, a la vez que facilita la gestión del proyecto.

Veremos cómo deben ser estos requisitos para elaborar el documento que los recopile.

Las dificultades en los proyectos de desarrollo

El desarrollo de software sigue siendo una actividad con gran componente **artesanal, científico** y **comunicativo**.

Es **artesanal** porque una parte importante se hace manualmente, una gran cantidad de las instrucciones de código se realizan letra a letra por parte del programador, si bien es cierto que los compiladores hacen una parte del trabajo, la más importante la realiza el programador, me refiero a la parte funcional de acuerdo con los requerimientos del cliente.

Es **científica** porque requiere de conocimientos técnicos acerca de cómo funciona el hardware sobre el que funcionará el producto desarrollado, ya sea un servidor, un móvil, una computadora o una red de comunicaciones.

Pero, también tiene un componente **comunicativo** porque el producto final dependerá mucho de la capacidad que tenga el usuario de definir adecuadamente sus necesidades y esto generalmente no sucede.

Lo más común es que el usuario perfile una aplicación, primero desde el punto de vista funcional y segundo con algunos requerimientos de diseño gráfico. Características como la seguridad y funcionalidad operativa consistente en que el producto desarrollado cuente con suficientes mensajes de alerta para indicar al usuario cuando se ha ingresado un dato con formato incorrecto o se ha realizado una violación a alguna regla operativa, que son importantísimas en cualquier software, tienden a ser omitidas por el usuario.

Además, generalmente las empresas no cuentan con un "dueño de producto" (*product owner*), es decir una persona experta en la solución buscada que conozca completamente las políticas, restricciones operativas y necesidades que debe cubrir el software, deja muchas de estas responsabilidades al grupo de desarrollo, que finalmente es el que menos conoce de estos aspectos del producto que se le ha solicitado desarrollar.

Esto finalmente genera dificultades, contratiempos y mutuos reclamos entre el grupo de desarrollo y los usuarios del software, provocando que la mayoría de los proyectos sean un continuo y constante flujo de quejas y descalificaciones entre los miembros del grupo de trabajo del proyecto.

¿Qué es la especificación de requisitos de software (SRS)?

La Especificación de Requisitos de Software (ERS o SRS por las siglas en inglés de Software Requirements Specification) es el documento que describe los requisitos de un sistema de software a desarrollar en 3 niveles:

- **Nivel superior**: estos son los requisitos comerciales de alto nivel. Describen los objetivos medibles del negocio, definen el propósito detrás del proyecto y alinean los objetivos del proyecto con los objetivos de las partes interesadas.
- **Nivel medio**: estos son los requisitos del usuario. Reflejan las necesidades y expectativas específicas del usuario, describen quién está usando el software y destacan las interacciones del usuario.
- **Nivel inferior**: estos especifican la funcionalidad del producto en términos tecnológicos. Identifican funciones, características, casos de uso y requisitos no funcionales, además de describir el proyecto como módulos funcionales más atributos no funcionales.

Al combinar las especificaciones de los tres niveles, se otorga a todas las partes interesadas una comprensión clara del proyecto y los entregables medibles, por eso este documento debe ser accesible para todos ellos: inversores, administradores de proyectos, programadores, diseñadores, etc.

Esto es especialmente importante si se está subcontratando el proyecto; un documento completo de SRS permite que una agencia de subcontratación comunique los requisitos de su proyecto de manera clara y precisa a su propio equipo.

Con la siguiente tabla obtendrás una idea más detallada de cómo un SRS mejora la comunicación y la colaboración entre las partes:

Miembro del grupo de trabajo	Actividad
Desarrollador de software	Deben utilizar el SRS para estimar con precisión el alcance del proyecto.
Gerente de proyecto	Deben utilizar el SRS para planificar el trabajo
Diseñador	Con base en el SRS, deben poder interpretar y definir las características visuales del proyecto.
Probador (tester)	Con base en el SRS deben tener la capacidad de planear los Casos de Prueba necesarios.
Usuarios y dueño del producto	Utilizan el SRS para validar al grupo de desarrollo que ha comprendido completamente desde el punto de vista funcional y operativo los requerimientos solicitados.

¿Por qué es importante definir y documentar la especificación de requisitos de software?

Hay muchas historias de terror sobre desarrolladores de software y clientes que chocan con los requisitos del proyecto, y una de las causas más comunes de tales disputas es el lenguaje ambiguo.

Términos vagos como "urgente", "pendiente", "lo antes posible", etc., pueden significar algo diferente para el cliente y para la empresa de desarrollo o entre el usuario y el grupo de desarrollo.

O, quizás, las partes involucradas tienen una impresión diferente de las funcionalidades prioritarias del producto. No es culpa de ninguna de las partes, sólo están hablando idiomas diferentes, en cierto sentido.

El cliente tiene una idea del producto, sus funciones y el aspecto que podría tener la interfaz de usuario. Pero los desarrolladores generalmente están pensando en el "detrás de escena" del proyecto, cómo escribir un código escalable y legible, garantizar la seguridad, hacer que una solución funcione, minimizar errores, etc.

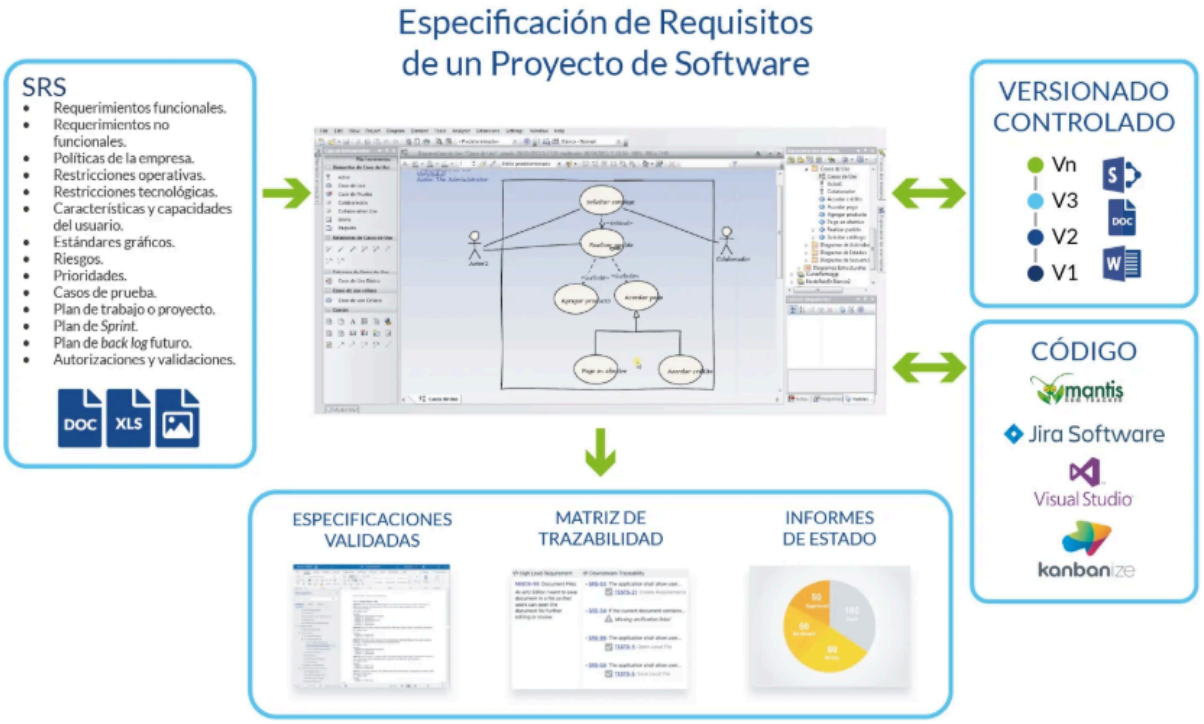
Cuando hay una desconexión entre ambos puntos de vista, puede generar trabajo adicional, dinero desperdiciado, estrés e incluso un producto final deficiente. Pero, afortunadamente, hay una manera de evitar tales malentendidos desde el principio: las especificaciones de los requisitos del software.

Partes de un buen documento de SRS

Para que un SRS sea útil para todas las partes involucradas, debe tener los siguientes elementos o secciones básicos:

1. **Impulsores comerciales**: incluye las razones por las que el cliente quiere que le fabriques el software. Es importante incluir la justificación o la fuerza impulsora detrás del nuevo sistema; esta información guía las decisiones que tomarán los analistas de negocios, los arquitectos de sistemas y los desarrolladores.
2. **Modelo de negocio**: ¿Qué modelo de negocio de cliente básico necesitará soportar el sistema? Incluye el contexto organizacional y comercial, las funciones comerciales clave, los diagramas de estado actuales y futuros y los diagramas de flujo de procesos.
3. **Casos de uso del sistema y del negocio**: por lo general, esta sección contendrá un diagrama de casos de uso (UML) que muestra con qué objetos externos principales interactuará el sistema.
4. **Requisitos técnicos**: en esta sección, el documento enumera los requisitos "no funcionales" que componen el entorno técnico.
5. **Cualidades del sistema**: enumerará los requisitos no funcionales que determinan la calidad del sistema.

6. **Restricciones y suposiciones**: esta sección incluye restricciones de software que requiere el cliente, que se pueden usar para excluir opciones de la consideración de desarrollo.
7. **Criterios de aceptación**: por último, este elemento ilustra cómo el cliente aprobará el sistema final.



¿Cómo deben ser los SRS y cómo validarlos?

Es importante aclarar que cuando se habla de grupo de trabajo se está incluyendo al usuario o al dueño del producto y no solamente hablo del grupo de técnico, por lo tanto, un buen documento SRS debe cumplir con las siguientes características:

1. **Corrección**: La información contenida es lo mismo para el usuario, el desarrollador, el diseñador y el probador. Después de estudiarlo ninguno se queda con un concepto diferente a lo que se desea con el proyecto de desarrollo.
2. **Exhaustividad**: No deja espacio para dudas para ninguno de los miembros del grupo de trabajo. Cualquier aspecto técnico y funcional queda aclarado y documentado en el SRS.
3. **Coherencia**: Para cada requerimiento existen reglas claras y no entran en conflicto con las reglas o la funcionalidad de otro requerimiento.

4. **Cero Ambigüedad**: Para cada tipo de actividad del Ciclo de Vida del Proyecto (desarrollador, diseñador, tester y usuario), el contenido no puede generar ideas generales, debe contener especificaciones concretas y completas desde todos los puntos de vista para cada una.
5. **Clasificación por importancia**: No todos los requerimientos son iguales, algunos son críticos, esto significa que si contienen algún error ya sea funcional o no funcional pueden provocar desde retrasos en la operación hasta impactos legales o financieros y por lo tanto los casos de prueba deben ser más exhaustivos. También sirve para determinar con qué tipo de requerimientos la liberación del código no puede ser considerada como realizada si no se han efectuado las pruebas correspondientes que acrediten con evidencia concreta que el requerimiento está completamente construido y probado y por lo tanto no implica un potencial riesgo operativo. Cada requerimiento debe contener la codificación de importancia que le corresponda.
6. **Modificabilidad**: El SRS es un documento vivo, esto significa que puede estar cambiando continuamente por lo tanto debe cumplir cabalmente con la práctica de Administración de Versiones o Versionado. Esto implica que cada nueva versión debe cumplir con las cinco características anteriores y debe ser aprobada por cada miembro del grupo de trabajo. Esto debido a que ciertos requerimientos pueden inducir, sobre todo de parte del usuario, ideas de mejora que antes no había concebido y por lo tanto desea que el nuevo software las contenga.
7. **Verificabilidad**: Todos los requerimientos deben ser verificados con evidencia documental u operativa concreta para todos los miembros del grupo de trabajo. Tanto en los informes de progreso como en el documento final de validación por parte del usuario.
8. **Trazabilidad**: Cualquier cambio realizado en el SRS y para cada requerimiento debe ser fácil y cabalmente evaluado atrás en el tiempo mediante la consulta de las diferentes versiones documentales del mismo.
9. **Comprobabilidad**: Cualquier requerimiento o solicitud debe contener el visto bueno o aprobación expresa, no tácita, de cada miembro del grupo de trabajo, tanto del solicitante como del responsable de ejecutarlo.
10. **Características del diseño**: El SRS debe contener todas las características estándar de diseño del producto de software y las características particulares de algún módulo en particular totalmente aprobadas por el usuario, considerando la política de imagen corporativa del cliente.
11. **Comprensible para el cliente**: El lenguaje utilizado debe ser lo suficientemente claro para el grupo de desarrollo, pero también para el cliente. El uso excesivo de tecnicismos o caló especializado alejará al cliente y no podrá realizar un comprometido análisis del requerimiento al no comprender en su lenguaje lo que se está documentando en el SRS.

12. **Nivel de abstracción adecuado**: El documento debe ser comprendido por todos los participantes en el grupo de trabajo sin importar su rol y tipo de participación en el proyecto.

Retroalimentación

Ejercicio 1: Conceptualización

Identificar conceptos clave de calidad del software.

Completar el siguiente cuadro comparativo.

Concepto	**Definición propia (por el alumno)**	**Ejemplo real**
Calidad del software		
Funcionalidad técnica		
Experiencia de usuario		
Especificación de requisitos (SRS)		
Requisito funcional		
Requisito no funcional		

Ejercicio 2: Análisis de caso

Identificar errores comunes en la definición de requisitos y sus consecuencias.

Caso práctico:

Una empresa contrata a un equipo de desarrollo para crear una app de reservas de citas médicas. El cliente dijo que quería que sea “rápido y amigable”. Luego de entregar el producto, el cliente se queja porque la interfaz no muestra alertas cuando el usuario ingresa datos incorrectos y porque el sistema no funciona bien en celulares.

Preguntas para resolver:

1. ¿Qué errores cometieron en la etapa de requisitos?

2. ¿Qué requisitos funcionales y no funcionales no fueron correctamente definidos?

3. ¿Cómo se hubiera evitado este conflicto mediante un buen SRS?

Ejercicio 3: Mapeo de estándares

Asociar normas con su propósito.

Completar la siguiente tabla con información del material.

Norma o estándar	**Propósito principal**	**Aplicación práctica en un proyecto de software**
CMMI		
ISO 12207		
ISO/IEC/IEEE 29119		
IEEE 730		

Ejercicio 4: Construcción de SRS (versión simplificada)

Redactar un mini-SRS para un sistema sencillo.

En parejas o grupos, redacten un **documento SRS simplificado** para un “Sistema de Registro de Visitas Escolares”.

Incluyan:

- Objetivo del sistema.
- Requisitos funcionales (mínimo 3).
- Requisitos no funcionales (mínimo 2).
- Casos de uso básicos.
- Criterios de aceptación.

Ejercicio 5: Debate guiado

Reflexionar sobre la importancia del lenguaje claro y no ambiguo en los requisitos.

Pregunta disparadora:

> ¿Quién debe adaptarse al lenguaje del otro: el cliente al técnico, o el técnico al cliente?

Dinámica:
Dividir la clase en dos grupos, cada uno defiende una postura. Luego se realiza una puesta en común sobre los riesgos de una comunicación inefectiva en proyectos de software.