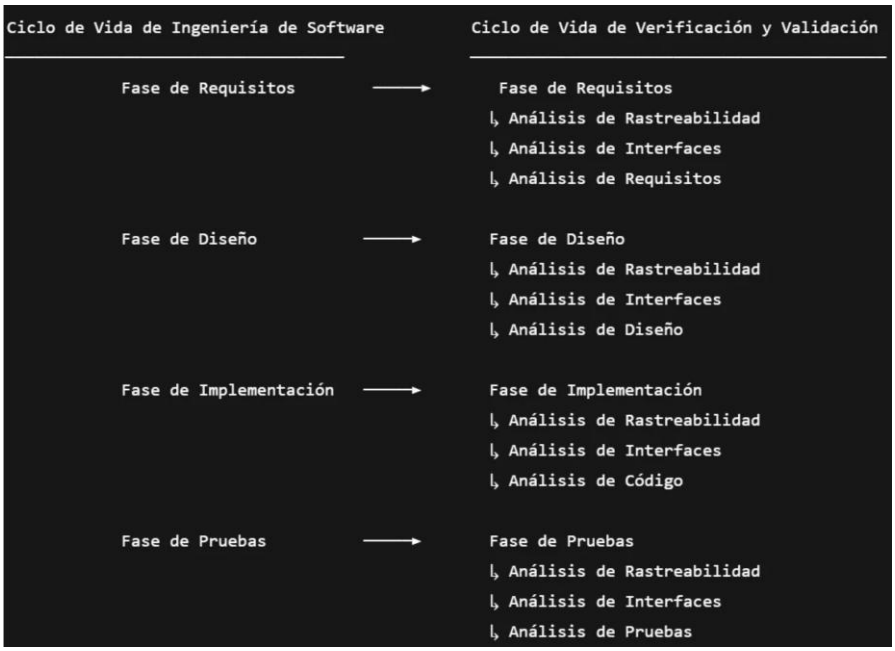
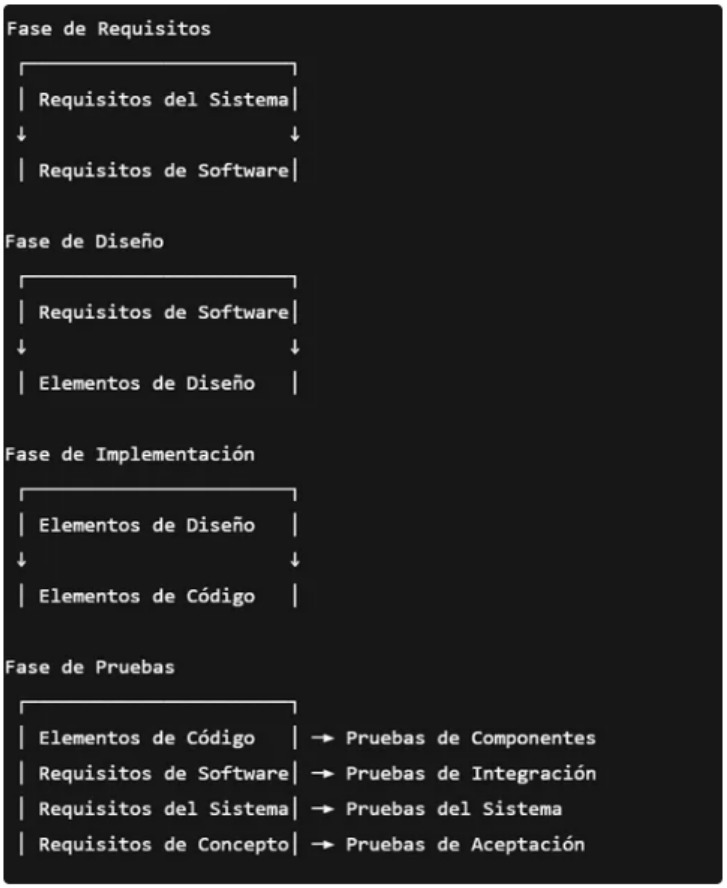
Ciclo de vida de V y V





¿Qué es?

Es una técnica dentro del ciclo de Verificación y Validación (V&V) para asegurar que cada fase del desarrollo de software (requisitos, diseño, implementación y pruebas) esté correctamente vinculada y alineada con las fases anteriores y siguientes.

Para cada una de las fases del ciclo de vida

El equipo de V&V quiere asegurarse de que los elementos de interfaz correctos:

hayan sido identificados, estén completamente definidos, se utilicen de forma consistente, mantengan las necesidades de rendimiento del sistema, puedan verificarse mediante pruebas

El equipo de V&V debe realizar las siguientes tareas genéricas:

Tarea 1: Identificar los datos que deben pasarse entre los módulos.
Tarea 2: Identificar las interfaces que deben manejar los datos.
Tarea 3: Comparar las interfaces definidas por V&V con las definidas por los desarrolladores y evaluar las inconsistencias.
Tarea 4: Analizar cada elemento de datos y determinar si es completamente definido.
Tarea 5: Grafique las ubicaciones donde se utilizan los elementos de datos y determine si se utilizan de manera consistente.

Tarea 6: Identificar las necesidades de rendimiento del sistema.
Tarea 7: Modelar y simular la comunicación entre interfaces para determinar si se mantienen estas necesidades de rendimiento.
Tarea 8: Desarrollar pruebas para las interfaces e identificar cuáles no son comprobables.

En resumen, el equipo de V&V cumplió con sus requisitos de análisis de interfaz y proporcionó información sobre la calidad de las interfaces al responder las siguientes preguntas:

1) ¿Los desarrolladores tienen las interfaces correctas definidas
2) ¿Se utilizan esas interfaces de forma consistente en todo el sistema?
3) ¿Esas interfaces mantienen las necesidades de rendimiento del sistema?
4) ¿Están completamente descritas esas interfaces?
5) ¿Es posible verificar esas interfaces mediante pruebas?

Métricas de V y V

Métricas de framework

Ayudas para la toma de decisiones que se utiliza para organizar, seleccionar, comunicar y evaluar los atributos de cálidas requeridos para un sistema de software

Factor de calidad

Un atributo del software orientado a la gestión que contribuye a su calidad

Subfactor de calidad

Descomposición de un factor de calidad en sus componentes técnicos

Métrica directa

Es una métrica que no depende de la medida de ningún otro atributo

Métricas predictiva

Métrica que se aplica durante el desarrollo y se utiliza para predecir los valores de un factor de calidad del software

Métricas de calidad del software

Una función cuyas entradas son datos del software y cuya salida es un único valor numérico que puede interpretarse como el grado en que el software posee un atributo dado que afecta a su calidad

Métricas de proceso

Métrica utilizada para medir las características de los métodos, técnicas y herramientas empleadas en el software

Métricas de productos

Métrica utilizada para medir esa característica de cualquier producto intermedio o final del proceso de desarrollo de software

¿Qué medir?

Esta pregunta se puede implementar mediante un proceso de cinco pasos:

1. Establecer requisitos de calidad del software
2. Identificar métricas de calidad del software
3. Implementar las métricas de calidad del software
4. Analizar los resultados de las métricas del software
5. Validar las métricas de calidad del software

Ejemplos de Métricas Directas

Subfactores de Calidad	Métricas Directas	Descripción
Independencia de hardware	Dependencias de hardware	Contar dependencias de hardware
Independencia de software	Dependencias de software	Contar dependencias de software
Instalabilidad	Tiempo de instalación	Medir tiempo de instalación
Reusabilidad	Otro software de aplicación puede usarse	Contar número de otros softwares que pueden ser o han sido usados
No deficiencia	Cobertura de pruebas, Inspección de código	Medir cobertura de pruebas, contar módulos inspeccionados
Tolerancia a errores	Integridad de datos, Recuperación de datos	Contar situaciones donde los datos del usuario se corrompen, recuperación
Disponibilidad	% de tiempo que el software está disponible	Tiempo disponible / Tiempo total disponible (expresado en porcentaje)
Completitud	Cobertura de pruebas	Medida de pares o cobertura de ramas

Subfactores de Calidad	Métricas Directas	Descripción
Corrección	Densidad de defectos	Contar defectos descubiertos en cada versión antes del lanzamiento
Seguridad	Integridad de datos, Seguridad del usuario	Contar corrupciones de datos, número de usuarios ilegales no prevenidos
Compatibilidad	Cambios ambientales	Número de variables de entorno que deben cambiar después de instalar
Interoperabilidad	Operabilidad en entornos mixtos	Número de entornos mixtos donde el software puede operar correctamente
Comprensibilidad	Tiempo de aprendizaje	Tiempo que tarda un nuevo usuario en aprender las características del software
Facilidad de aprendizaje	Tiempo de aprendizaje	Tiempo que tarda un nuevo usuario en aprender funciones básicas
Operabilidad	Factores humanos	Nº de comentarios negativos de nuevos usuarios sobre ergonomía, factores humanos, etc.
Comunicatividad	Factores humanos	Nº de comentarios negativos de nuevos usuarios sobre ergonomía, factores humanos, etc.

Validar las métricas de calidad del software

El propósito de validar las métricas es ganar confianza en que los números reflejan la realidad

Complejidad

La complejidad es una métrica directa que puede asociarse con el subfactor de calidad corregibilidad y el factor de calidad mantenibilidad.

Métricas de defectos

Se recopilan del resumen de inspección

Las métricas de defectos respaldan las actividades de V&V del software al permitir el **seguimiento de los defectos por módulo**

Esto puede identificar módulos que pueden ser candidatos para rediseño o requerir pruebas adicionales

Métricas de Producto:

- Número de defectos encontrados durante pruebas funcionales: Se pueden registrar por módulo (por ejemplo: ventas, inventario, usuarios).
- Número de líneas de código modificadas o creadas: Extraído del control de versiones (Git).
- Número de reportes generados correctamente en PDF.
- Número de módulos desarrollados vs planificados.
- Complejidad promedio de módulos: Si se usa una herramienta como SonarQube o Code Metrics en Visual Studio.

Métricas de Proceso:

Las métricas de proceso tienen como objetivo reflejar la eficacia de sus procesos.

Al recopilar estas medidas y analizar los resultados en varios proyectos, puede identificar tendencias que deberían conducir a mejoras en los procesos

- Tiempo promedio de corrección de defectos (find-fix cycle).
- Horas-hombre por revisión de código.
- Promedio de defectos encontrados por revisión.
- % de módulos inspeccionados antes de liberarse.
- Tiempo entre detección y corrección del defecto.

Tipos de defectos

- Basado en las prácticas típicas de V&V, y los elementos de tu proyecto, podrías usar la siguiente clasificación:

- Defectos de lógica: errores en cálculos o validaciones incorrectas (por ejemplo, stock negativo permitido).
- Defectos de interfaz: elementos mal alineados, etiquetas incorrectas, flujo de navegación roto.
- Defectos de definición de datos: uso de datos inválidos o ausentes en formularios (por ejemplo, campos obligatorios sin validar).
- Defectos de documentación: inconsistencias entre manual de usuario, requerimientos y el comportamiento real.
- Defectos de integración: fallas al conectar módulos entre sí (ej. ventas no descuenta inventario).
- Defectos de seguridad: accesos sin autenticación, errores en la gestión de roles.
- Defectos de rendimiento: tiempos de respuesta superiores a los 2 segundos (según tu RNF04).

El siguiente conjunto de atributos, sugerido por Humphrey, debe ser considerado:

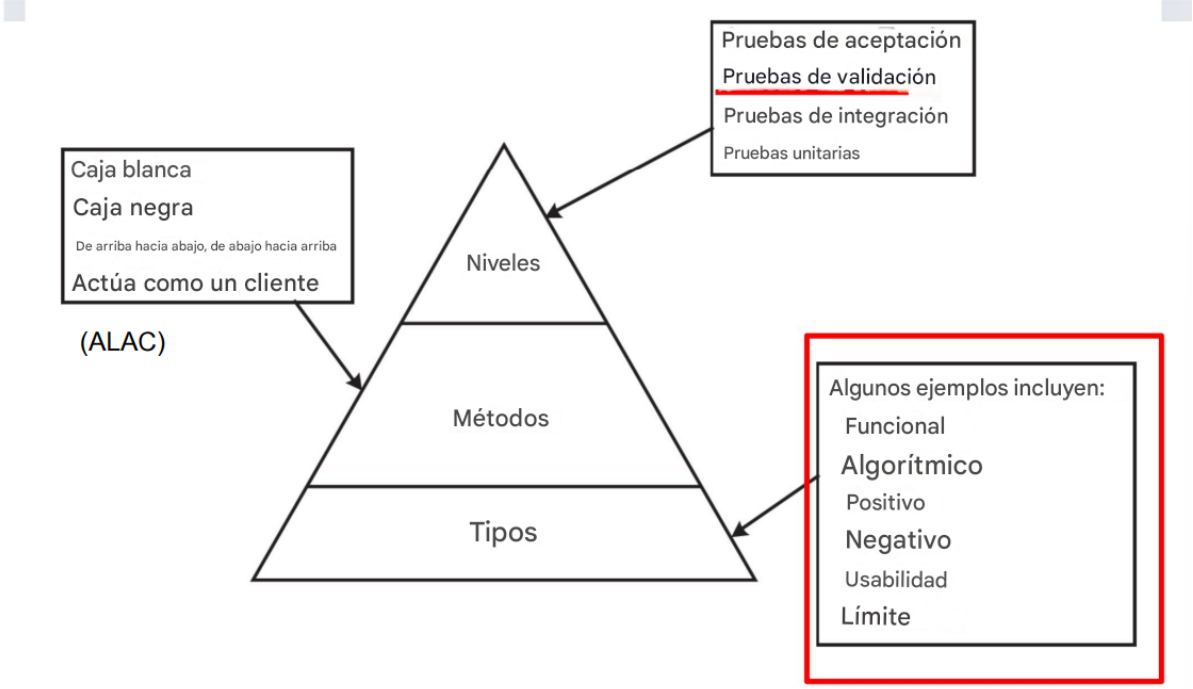
- Las medidas **deben ser robustas**.
- Las medidas **deben sugerir una norma**.
- Las medidas deben relacionarse con propiedades específicas del producto y del proceso.
- Las medidas deben sugerir una estrategia de mejora.
- Las medidas deben ser un resultado natural del proceso de desarrollo de software.
- Las medidas **deben ser simples**.
- Las medidas **deben ser predecibles y rastreables**.
- Las medidas **no deben utilizarse como parte de la evaluación del desempeño de una persona**.

- Es muy importante reconocer la diferencia entre cambiar y mejorar:

Mejorar se basa en la medición, mientras que **cambiar se basa en la percepción**.

La manera en que sabes que un cambio es una mejora es a través de la *medición* [Rakitin].

Técnicas de validación



Validación

El proceso de valuar un sistema o componente durante o al final del proceso de desarrollo para determinar si satisface los requisitos especificados

Verificacion y Validacion

proceso de determinar si los requisitos para un sistema o componente son completos y correctos, los productos de cada fase de desarrollo cumplen los requisitos o condiciones impuestos por la fase anterior y el sistema o componente final cumple con los requisitos especificados cf. verificación y validación independiente

Evaluación

determinación sistemática del grado en que una entidad cumple sus criterios específicos

Validación

Confirmación, mediante la presentación de evidencia objetiva, de que se han cumplido los requisitos para un uso o aplicación específicos previstos. 5. La garantía de que un producto, servicio o sistema satisface las necesidades del cliente y otras partes interesadas identificadas. A menudo implica la aceptación y la idoneidad de los clientes externos.

Verificacion

proceso de evaluación de un sistema o componente para determinar si los productos de una fase de desarrollo dada satisfacen las condiciones impuestas al inicio de esa fase

Validación de pruebas

Determinar si el software cumple con todos sus requisitos tal como se definen en el SRS

Pruebas de regresión

implican volver a ejecutar de forma selectiva pruebas de validación para garantizar que las correcciones de errores no hayan introducido nuevos errores

Pruebas alfa y beta

Algunas organizaciones involucran activamente a los clientes en las pruebas al proporcionar software preliminar para que el cliente lo evalúe a través de lo que comúnmente se denominan alfa y beta

Pruebas de aceptación

son similares a las pruebas de validación, con una diferencia: el cliente participa activamente

Pueden ser una repetición (o un subconjunto) de las mismas pruebas utilizadas para las pruebas de validación o pueden emplear pruebas desarrolladas íntegramente por los cliente

Las pruebas de regresión se realizan para determinar si el software aún cumple con todos sus requisitos a la luz de los cambios y modificaciones realizados en el software

Implica repetir selectivamente pruebas de validación existentes, no desarrollar nuevas pruebas

Pruebas alfa y beta

Para que las pruebas alfa y beta sean más efectivas, debe proporcionar a sus clientes un esquema de los aspectos en los que desea que se concentren y escenarios de prueba específicos para que ejecuten.

Niveles de prueba y métodos de prueba

Nivel de prueba	Objetivos	Realizado por	Entorno de prueba	Métodos de prueba
Unidad	Detectar errores en la lógica, los datos y los algoritmos en módulos individuales.	Ingenieros de software	Aislado. Pueden requerirse stubs y andamiaje.	Caja blanca
Integración	Detectar errores en las interfaces entre módulos.	Ingenieros de software	Aislado o simulado. Se requieren stubs y andamiaje.	Caja blanca De arriba a abajo y de abajo a arriba
Validación	Determinar si el control de calidad del software cumple con el SRS.		Real	Funcional y ALAC
Regresión	Determinar si el software aún cumple con el SRS a la luz de los cambios.		Real	Funcional y ALAC
Aceptación	Determinar si el software cumple con los requisitos del cliente.	Cliente, control de calidad o equipo del proyecto	Real (generalmente en las instalaciones del cliente)	Funcional y ALAC

Algunos de los tipos de pruebas más comunes son:

Funcional.
Pruebas diseñadas para determinar si funciones o características específicas funcionan según lo especificado.
Algorítmica. Pruebas diseñadas para determinar si se han implementado algoritmos específicos correctamente.

Pruebas positivas.
Pruebas diseñadas para determinar si una característica produce resultados consistentes con los requisitos establecidos cuando el software se utiliza correctamente.

Pruebas negativas.
Pruebas diseñadas para determinar si el software se comporta razonablemente ante entradas no válidas o acciones inesperadas del operador.

Pruebas de usabilidad.
Pruebas que ejercitan características específicas de la interfaz de usuario para determinar si el software se comporta como lo esperarían usuarios capacitados o no capacitados.

Pruebas de límites.
Pruebas que ejercitan limitaciones específicas del producto, como valores mínimos y máximos, para determinar si el software se comporta razonablemente.

Pruebas de inicio/apagado.
Pruebas diseñadas para determinar si las funciones de inicio y apagado tienen se ha implementado correctamente.

Pruebas de configuración.
Pruebas diseñadas para determinar si una característica produce resultados consistentes con los requisitos establecidos cuando el software se utiliza correctamente.

Pruebas de plataforma.
Pruebas diseñadas para determinar si el software funciona correctamente en todos los Plataformas/sistemas operativos compatibles.

Pruebas de carga/estrés.
Pruebas que someten al producto a una carga establecida o esperada.

Validacion formal

SOLO SE REALIZAN CAMBIOS AL CODIGO EN RESPUESTA A ERRORES INFORMADO DUARANTZE LAS PRUEBAS DE VALIDACION; NO SE AGREGAN NUEVAS CARACTERISTICAS

Proporciona una oportunidad para que se desarrollen y depuren pruebas de validación en las primeras etapas del proceso de desarrollo de software; Proporciona retroalimentación temprana a los ingenieros de software

Validacion informal

LOS DESARROLLADORES PUEDEN REALIZAR CUALQUIER CAMBION NECESARIO PARA QUE CUMPLA CON EL SRS

Gestion del compromiso y riesgo

Haciendo compromisos

A un nivel personal, un compromiso es un acuerdo de hacer algo. Los compromisos no se deben tomar a la ligera

Puntos a considerar al hacer un compromiso, según Humphrey:

1. La persona que hace el compromiso lo hace voluntariamente
2. El compromiso no se hace a la ligera, hay que considerar el trabajo involucrando, recursos y organización
3. Hay un acuerdo entre las partes sobre las que debe hacerse, quien lo debe hacer y cuando
4. El compromiso se declara abiertamente y públicamente
5. La persona responsable intenta cumplir el compromiso, aún si necesitan ayuda
6. Antes de la fecha de compromiso, si está claro que no se podrá cumplir, se da un aviso previo y se replantea un nuevo compromiso

Riesgo

Son eventos que pueden generar un impacto negativo en un proyecto y/o producto

Después de identificar cientos de proyectos, Jones identifico 10 riesgos serios de software

1. Métricas inadecuadas
2. Medidas inadecuadas
3. Presión excesiva del cronograma
4. Malas prácticas de administración
5. Estimación de costos inadecuados
6. Síndrome de la bala de plata (Creer que hay una solución mágica para todo)
7. Requisitos progresivos de los usuarios
8. Baja calidad
9. Baja productividad
10. Cancelación de proyecto

Metodos formales

Un método formal es una técnica basada en matemáticas, utilizada para describir sistemas de hardware o software [Wing, Jeannette M., 1990].

Especificación formal. Se utiliza una notación matemática para proporcionar una descripción precisa de lo que debe hacer un pro

Verificación formal. Se utilizan reglas precisas para demostrar matemáticamente

que un programa satisface una específica formal

10 mandamientos de los métodos formales

1. Elegirás la notación apropiada.
2. Formalizarás pero no en exceso.
3. Estimarás los costos (entrenamiento, adquisición de herramientas, consultores).
4. Tendrás un experto en métodos formales a tu disposición.
5. No abandonarás los métodos tradicionales de desarrollo.
6. Documentarás suficientemente. Es recomendable comentarios en lenguaje natural.
7. No comprometerás los estándares de calidad.
8. No serás dogmático. No hay garantía de corrección.
9. Probarás, probarás y probarás de nuevo.
10. Reutilizarás. (Los MF son un enfoque apropiado cuando se han de crear componentes).