Contents

Clase 9: Técnicas de Validación de Requisitos de Calidad

Objetivos de la clase:

- Comprender en detalle diversas técnicas de validación de requisitos de calidad.
- Aplicar técnicas de validación a diferentes tipos de requisitos.
- Analizar los pros y contras de cada técnica de validación.
- Saber cuándo y cómo elegir la técnica de validación más apropiada.

Contenido Teórico Detallado:

En la clase anterior, introdujimos la Verificación y Validación (V&V) de requisitos de calidad y nos centramos en las técnicas de verificación. Ahora nos centraremos en la Validación.

La **Validación** tiene como objetivo asegurar que los requisitos de calidad, una vez implementados en el software, realmente satisfacen las necesidades y expectativas de los usuarios y stakeholders. A diferencia de la verificación, que se centra en la corrección de la implementación, la validación se centra en la adecuación del producto.

A continuación, se describen algunas de las técnicas de validación más comunes:

1. **Revisiones de Stakeholders:**
   - **Descripción:** Presentar los requisitos de calidad (o prototipos del software que implementan esos requisitos) a los stakeholders (usuarios, clientes, expertos del dominio) para obtener su feedback y aprobación.
   - **Proceso:**
     - Planificar la revisión, identificando los stakeholders relevantes y los objetivos de la revisión.
     - Preparar los materiales de revisión, como documentos de requisitos, prototipos o simulaciones.
     - Realizar la revisión, fomentando la participación activa y la discusión abierta.
     - Documentar los resultados de la revisión, incluyendo los comentarios, sugerencias y decisiones tomadas.
   - **Ventajas:** Identifica problemas de usabilidad, rendimiento u otros aspectos que no se detectan internamente. Obtiene la aprobación de los stakeholders, aumentando la confianza en el producto final.
   - **Desventajas:** Requiere la disponibilidad y participación activa de los stakeholders, lo que puede ser costoso y consumir tiempo. La retroalimentación puede ser subjetiva y difícil de interpretar.
   - **Cuándo usar:** Cuando se necesita la opinión experta de los usuarios y clientes para garantizar que el software satisfaga sus necesidades.
   - **Prototipado:**
   - **Descripción:** Crear versiones preliminares del software (prototipos) para evaluar la viabilidad de los requisitos y obtener feedback temprano de los usuarios.
   - **Tipos de prototipos:**
     - *Prototipos exploratorios:* Se utilizan para comprender mejor los requisitos y explorar diferentes alternativas de diseño.
     - *Prototipos evolutivos:* Se construyen de forma incremental, añadiendo funcionalidades gradualmente.
     - *Prototipos desechables:* Se crean rápidamente para validar un concepto y luego se descartan.
   - **Proceso:**

  - Definir los objetivos del prototipo.
  - Diseñar y construir el prototipo.
  - Evaluar el prototipo con los usuarios.
  - Refinar los requisitos en función del feedback obtenido.
- **Ventajas:** Permite identificar problemas y realizar ajustes antes de la implementación completa, reduciendo los riesgos. Facilita la comunicación y la colaboración entre desarrolladores y usuarios.
- **Desventajas:** Puede ser costoso y consumir tiempo, especialmente si se crean prototipos complejos. Los usuarios pueden tener expectativas poco realistas sobre el prototipo.
- **Cuándo usar:** Cuando los requisitos son complejos, ambiguos o poco claros, o cuando se necesita validar la viabilidad de una nueva tecnología o enfoque.
- **Casos de Uso y Escenarios:**
- **Descripción:** Desarrollar casos de uso y escenarios que describan cómo los usuarios interactúan con el software para lograr objetivos específicos.
- **Proceso:**
  - Identificar los actores (usuarios, sistemas externos) y los objetivos que desean alcanzar.
  - Describir los pasos que los actores realizan para lograr sus objetivos, incluyendo los flujos principales y alternativos.
  - Validar los casos de uso y escenarios con los stakeholders para asegurar que reflejen sus necesidades y expectativas.
- **Ventajas:** Ayuda a comprender el comportamiento del sistema desde la perspectiva del usuario. Facilita la identificación de requisitos faltantes o inconsistentes.
- **Desventajas:** Puede ser difícil crear casos de uso y escenarios completos y realistas. Los casos de uso y escenarios pueden no cubrir todos los posibles casos de uso del software.
- **Cuándo usar:** Cuando se necesita comprender y validar el comportamiento del sistema desde la perspectiva del usuario.
- **Pruebas de Aceptación del Usuario (UAT):**
- **Descripción:** Realizar pruebas con usuarios reales en un entorno lo más similar posible al de producción para validar que el software cumple con sus requisitos y expectativas.
- **Proceso:**
  - Planificar las pruebas, definiendo los objetivos, el alcance y los criterios de aceptación.
  - Preparar los datos de prueba y el entorno de pruebas.
  - Ejecutar las pruebas con los usuarios.
  - Documentar los resultados de las pruebas, incluyendo los defectos encontrados.
  - Corregir los defectos y volver a ejecutar las pruebas hasta que se cumplan los criterios de aceptación.
- **Ventajas:** Valida que el software satisface las necesidades y expectativas de los usuarios en un entorno real. Aumenta la confianza en el producto final.
- **Desventajas:** Puede ser costoso y consumir tiempo, especialmente si se requiere la participación de un gran número de usuarios. Las pruebas pueden no cubrir todos los posibles casos de uso del software.
- **Cuándo usar:** Antes de la liberación del software para asegurar que cumple con los requisitos y expectativas de los usuarios.
- **Simulaciones y Modelado:**

- **Descripción:** Utilizar simulaciones y modelos para predecir el comportamiento del software en diferentes escenarios y condiciones, y validar que cumple con los requisitos de rendimiento, seguridad u otros aspectos.
- **Proceso:**
  - Definir los objetivos de la simulación o el modelo.
  - Construir el modelo o la simulación.
  - Ejecutar la simulación o el modelo.
  - Analizar los resultados y validar que cumplen con los requisitos.
- **Ventajas:** Permite evaluar el comportamiento del software en escenarios que son difíciles o imposibles de probar en un entorno real. Ayuda a identificar problemas potenciales antes de la implementación.
- **Desventajas:** Puede ser difícil construir simulaciones y modelos precisos y realistas. Los resultados de las simulaciones y modelos pueden no ser siempre precisos.
- **Cuándo usar:** Cuando se necesita evaluar el comportamiento del software en escenarios complejos o críticos, como sistemas de control o aplicaciones de seguridad.

Ejemplos y Casos de Estudio:

- **Caso de Estudio: Aplicación de Banca Móvil**
  - **Requisito de Usabilidad:** "El usuario debe poder realizar una transferencia de fondos en menos de 3 clics."
  - **Técnica de Validación:** Pruebas de Usabilidad con usuarios reales. Se les pide que realicen una transferencia y se mide el número de clics necesarios.
  - **Resultado:** Se observa que, en promedio, los usuarios necesitan 5 clics para completar la transferencia. Se rediseña la interfaz para reducir el número de clics a 3.
- **Caso de Estudio: Sistema de Control de Tráfico Aéreo**
  - **Requisito de Rendimiento:** "El sistema debe poder procesar hasta 1000 vuelos simultáneamente sin degradación del rendimiento."
  - **Técnica de Validación:** Simulación. Se simula un escenario con 1000 vuelos y se mide el tiempo de respuesta del sistema.
  - **Resultado:** Se observa que el tiempo de respuesta se degrada significativamente cuando se alcanzan los 800 vuelos. Se optimiza el código y la infraestructura para mejorar el rendimiento.
- **Caso de Estudio: Aplicación de Comercio Electrónico**
  - **Requisito de Seguridad:** "La información de las tarjetas de crédito de los usuarios debe estar protegida contra accesos no autorizados."
  - **Técnica de Validación:** Revisión de Seguridad con expertos en seguridad. Se revisa el código y la infraestructura para identificar vulnerabilidades potenciales.
  - **Resultado:** Se identifican varias vulnerabilidades, como la falta de encriptación en ciertas áreas y la posibilidad de ataques de inyección SQL. Se corrigen las vulnerabilidades y se realizan pruebas de penetración para verificar la efectividad de las medidas de seguridad.

Problemas Prácticos o Ejercicios con Soluciones:

1. **Ejercicio:** Para una aplicación de gestión de proyectos, propón dos requisitos de calidad (uno no funcional y uno funcional) y la técnica de validación más adecuada para cada uno, justificando tu elección.
   - **Solución:**
     - *Requisito Funcional:* "El usuario debe poder generar un informe del progreso del proyecto en formato PDF." - Técnica de Validación: Revisiones de Stakeholders (mostrar el informe generado a los gestores de proyecto y obtener su aprobación).
     - *Requisito No Funcional (Rendimiento):* "La aplicación debe cargar la lista de tareas en menos de 3 segundos." - Técnica de Validación: Pruebas de Rendimiento (medir el tiempo de carga de la lista de tareas en diferentes condiciones de carga).

2. **Ejercicio:** Identifica las ventajas y desventajas de utilizar Prototipado como técnica de validación para una aplicación móvil.
   - **Solución:**
     - *Ventajas:* Permite obtener feedback temprano de los usuarios sobre la usabilidad y la funcionalidad. Ayuda a identificar problemas y realizar ajustes antes de la implementación completa. Facilita la comunicación y la colaboración entre desarrolladores y usuarios.
     - *Desventajas:* Puede ser costoso y consumir tiempo. Los usuarios pueden tener expectativas poco realistas sobre el prototipo. Puede ser difícil crear prototipos que sean representativos del producto final.
3. **Ejercicio:** ¿En qué situación sería más apropiado usar Simulaciones y Modelado en lugar de Pruebas de Aceptación del Usuario (UAT) para validar un requisito de calidad?
   - **Solución:** Simulaciones y Modelado serían más apropiados cuando el escenario a validar es difícil, costoso o peligroso de probar en un entorno real. Por ejemplo, para validar el requisito de "El sistema debe ser capaz de resistir un ataque DDoS," sería más apropiado utilizar una simulación que intentar realizar un ataque real en un entorno de producción. También serían apropiadas para probar escenarios con un gran número de usuarios concurrentes.

Materiales Complementarios Recomendados:

- Libro: "Software Engineering" de Ian Sommerville (Capítulo sobre pruebas y validación).
- Artículo: "Validating Software Requirements" de IEEE Software.
- Curso online: "Software Testing and Validation" en Coursera o edX.
- Estándar ISO/IEC 29119 (Software Testing).
- Ejemplos de planes de pruebas de aceptación.