1. Explica que son la verificación y la validación en el contexto del desarrollo de software. Incluye las diferencias clave entre ambos conceptos y porque son importantes.

Verificación: el producto fue desarrollado de acuerdo a las especificaciones técnicas.

Validación: el producto fue desarrollado en cumplimiento con los requisitos del usuario final.

Diferencias clave: el enfoque es distinto, la verificación se concentra en las especificaciones del sistema y la validación se concentra en el usuario final.

1. Explica cómo y en qué momentos del ciclo de vida del software se aplican estas prácticas.

Verificación: se lleva a cabo durante las fases de prueba, para asegurar que el sistema se construya de acuerdo a las especificaciones.

Validación: se lleva a cabo durante las fases de prueba, para asegurar que el sistema cumpla con las expectativas del cliente.

1. Investiga y describe al menos 3 técnicas de verificación utilizadas en el desarrollo de software(por ejemplo, revisiones de código, pruebas unitarias, análisis estático). Explica como se llevan a cabo y que beneficios ofrecen.
2. métodos
3. Review: verificar que los requisitos están completos, los diseños cumplan con los requisitos y el código no tenga errores. Se lleva a cabo con 1 o 2 miembros, es menos costoso y depende de la habilidad de los desarrolladores.
4. Inspecciones: dos tipos, de código y de diseño. La inspección de código es el proceso donde un equipo revisa el código fuente para encontrar errores. Y la inspección de diseño es la revisión formal de documentos de diseño. Se lleva a cabo por 3 a 6 miembros, donde todos los participantes deben estar preparados, y al final se realiza un reporte preparado por el moderador. Las ventajas es que este método es efectivo, pero puede detectar fallas, aunque su desventaja es que es caro y requiere desarrolladores habilidosos.
5. Pruebas unitarias: el proceso en el que se prueba la unidad funcional de código mas pequeña. Ayudan a garantizar la calidad del código, Una practica recomendada es escribir el software como unidades pequeñas y funcionales, y luego escribir una prueba unitaria para cada unidad de código. Luego, ejecute ese código de prueba de forma automática cada vez que realice cambios en el código de software. Así, si una prueba falla, puede aislar con rapidez el área del código que tiene el error. Las pruebas unitarias imponen paradigmas de pensamiento modular y mejoran la cobertura y calidad de las pruebas, Las pruebas unitarias automatizadas permiten que usted o sus desarrolladores dispongan de mas tiempo para centrarse en la programación.
6. Investiga y describe al menos 3 técnicas de validación utilizadas en el desarrollo de software (por ejemplo, pruebas de aceptación, pruebas del sistema, pruebas de usabilidad). Explica como se llevan a cabo y que beneficios ofrecen.
7. Prueba de aceptación de usuario: los usuarios finales prueban la aplicación en un entorno de prueba. Su objetivo es comprobar que el sistema resuelve los problemas de usuarios. Aumenta la confianza del usuario al software.
8. Prueba de usabilidad: evalúa la facilidad de uso de la aplicación mediante pruebas con usuarios reales o simulados.
9. Pruebas de sistema: evalúa la totalidad para asegurar que todos los componentes funcionen juntos según lo previsto
10. Busque un ejemplo de un caso práctico.

Ejemplo de escenario real: aplicación móvil. Su proceso de verificación implica asegurar que los elementos de la aplicación tengan sentido respecto a sus especificaciones de diseño. Incluye modelos de información, maquetas de aplicaciones, verificaciones de los esquemas, consistencia de diseño, revisiones y una lista de verificación para asegurar el cumplimiento.

Para un ejemplo de validación, un proceso de compra de plataforma de comercio electrónico, los testers ejecutarían casos de prueba que simulan interacciones de usuarios de mundo real, aplicación de códigos de descuento, introducción de datos de tarjeta de crédito, uso de herramientas de análisis dinámicos para monitorear el rendimiento del sistema, etc.