Aseguramiento de Calidad de Software
IDS-308

Ingeniería de Software

Tema 2

Inspecciones y Revisiones:

Técnicas para llevar a cabo Inspecciones y Revisiones

Tópicos

Objetivos

Introducción

Consideraciones Importantes

Propósito de las Inspecciones y Revisiones del Software

Ténicas Estáticas y Dinámicas

Ventajas y Desventajas

Inspecciones y Revisiones del Código Fuente

Objetivos

- Conocer las técnicas para llevar a cabo inspecciones y revisiones

- Analizar las técnicas específicas que aplican para las revisiones de productos de trabajos, componentes y pruebas de sistemas

Introducción

Las inspecciones y revisiones se inician con los requerimientos y continúan con las revisiones del diseño y las inspecciones del código hasta la prueba del producto.

Consideraciones Importantes

Es fácil cometer errores y omisiones durante la fase de análisis de requerimientos del sistema y, en tales casos, el software final no cumplirá la expectativas de los clientes.

En la realidad, en las revisiones e inspecciones de los requerimientos, código, productos de trabajo en general no se puede descubrir todos los problemas que presenta la aplicación.

Algunos defectos en los requerimientos y aplicación sólo pueden descubrirse cuando la implementación del sistema es completada.

Propósito de las Inspecciones y Revisiones del Software

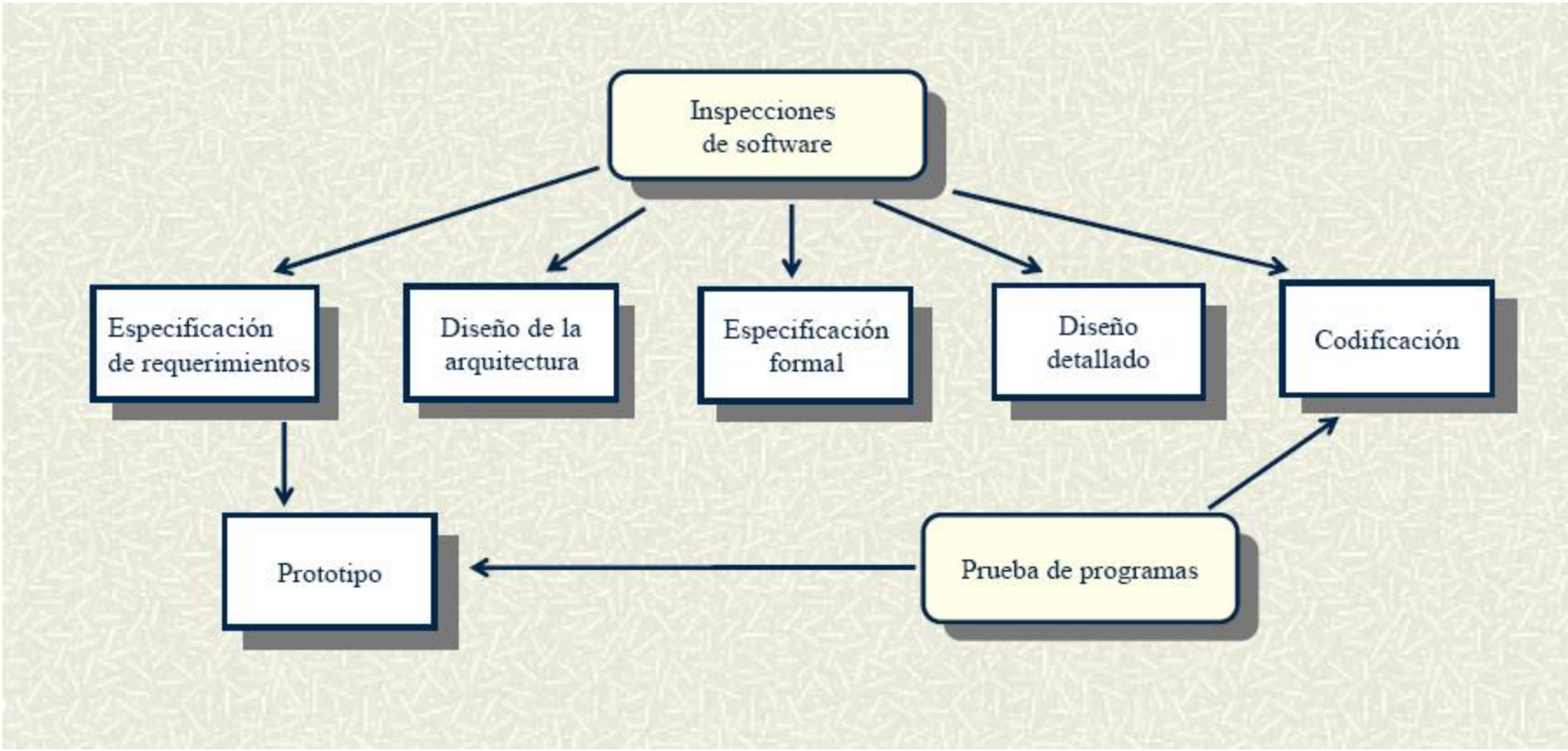
Analizan y comprueban las representaciones del sistema como el documento de requerimientos, los diagramas de diseño y el código fuente del programa. Se aplica a todas las etapas del proceso de desarrollo.

Las inspecciones se complementan con algún tipo de análisis automático del texto fuente o de los documentos asociados.

Las inspecciones del software y los análisis automatizados son técnicas de verificación y validación estáticas puesto que no requieren que el sistema se ejecute.

Asegurar que el software cumple las expectativas del cliente; así como comprobar que el sistema está acorde con su especificación.

Inspecciones y Revisiones Estáticas y Dinámicas



Ventajas y Desventajas

Ventajas:

- Las inspecciones y revisiones de software se pueden utilizar en todas las etapas del proceso de desarrollo, mientras que las técnicas de pruebas sólo se pueden usar cuando está disponible un prototipo, código fuente o ejecutable.

- Las inspecciones se pueden aplicar a la detección de fallos en cualquiera de los documentos generados.

- Permite utilizar el conocimiento del dominio y del lenguaje, que determinan los principales tipos de fallos que se suelen cometer.

Ventajas y Desventajas

Desventajas:

- Sólo pueden comprobar la correspondencia entre un producto de trabajo y su especificación y no puede probar que el software es de utilidad operacional, y mucho menos que las características no funcionales del software son las correctas.

- No detectan fallos a nivel de sistema, como es posible lograr en el proceso de pruebas.

- Por lo tanto, para validar un sistema o software, siempre se requieren llevar a cabo ciertas pruebas.

- No son útiles para la detección de niveles de fiabilidad y evaluación de fallos no funcionales.

Inspecciones y Revisiones del Código Fuente

Los analizadores estáticos de programas son herramientas de software que rastrean el texto fuente de un programa, en busca de errores no detectados por el compilador.

En algunos lenguajes de programación no son muy útiles pues el compilador ofrece una gran información acerca de posibles errores (incluso en ejecución).

Inspecciones y Revisiones del Código Fuente

Aspectos habitualmente analizados son:

- **Análisis de flujo de control:**

Identifica y señala los bucles con múltiples puntos de salida y las secciones de código no alcanzable.

- **Análisis de utilización de datos:**

Señala como se utilizan las variables del programa: Variables sin utilización previa, que se declaran dos veces, otras declaradas y nunca utilizadas. Esto es Condiciones lógicas con valor invariante, etc.

- **Análisis de interfaces:**

Verifica la declaración de las operaciones y su invocación. Esto es inútil en lenguajes tipificados como fuertes o robustos; por ejemplo Java.

- **Análisis de la trayectoria:**

Identifica todas las posibles trayectorias del programa y presenta las sentencias ejecutadas en cada trayectoria.

Bibliografía

Testing Computer Software, Kaner, C 3rd Edition. Hungry Minds Inc., 2008

Software Quality Engineering: Testing, Quality Assurance, and Quantifiable, Tian, J.

Ingeniería de Software, Ian Sommerville. 7ma Edición.

RUP: Rational Unified Process