DuocUC® INFORMÁTICA Y TELECOMUNICACIONES

Importancia y técnicas de inspección de código

Calidad de Software - CSY4111

CONTENIDO

DuocUC®

Refrescando conocimiento

DuocUC®

REFRESCANDO CONOCIMIENTO

El propósito principal de las pruebas de seguridad es detectar vulnerabilidades y luego repararlas.

Ayuda a impulsar el sistema actual y asegurarse de que el sistema pueda funcionar durante un tiempo prolongado. Para notar lagunas que provocarán la pérdida de información vital.

Es importante mencionar que estas pruebas se deben realizar siempre con el consentimiento del cliente.

REFRESCANDO CONOCIMIENTO

Las pruebas de seguridad son fundamentales para garantizar la protección de los datos sensibles y la integridad del sistema.

Algunos de los beneficios clave de realizar pruebas de seguridad son:

1.- Identificación de vulnerabilidades:

Las pruebas de seguridad ayudan a identificar y corregir vulnerabilidades en el software antes de que sean explotadas por atacantes.

Esto reduce el riesgo de brechas de seguridad y la exposición de datos sensibles.

01

INTRODUCCIÓN A LAS PRUEBAS DE CÓDIGO

DuocUC®

INTRODUCCIÓN A LAS PRUEBAS DE CÓDIGO

Las pruebas de inspección de código son un proceso de revisión sistemática y exhaustiva del código fuente de un programa para identificar errores, vulnerabilidades, malas prácticas y oportunidades de mejora.

Estas pruebas se basan en el análisis estático del código, lo que significa que se examina el código sin ejecutarlo.

El objetivo principal de las pruebas de inspección de código es garantizar la calidad y la robustez del software.

INTRODUCCIÓN A LAS PRUEBAS DE CÓDIGO

Las inspecciones son una técnica formal de revisión de código cuyo objetivo principal es detectar e identificar anomalías en un producto de software antes de lanzarlo a producción.

Consiste en el examen sistematizado por parte de pares del trabajo elaborado por una persona.

En el caso de las inspecciones se buscan errores comunes, por lo cual se examina para detectar la presencia de errores más que simular la ejecución.

Por tanto, resulta necesaria una lista de errores a detectar, en la cual se incluyan los errores de programación clásica.

INTRODUCCIÓN A LAS PRUEBAS DE CÓDIGO

Existen fundamentalmente dos criterios de salidas.

El primero de ellos consiste en aprobar el producto, aunque se hayan detectado anomalías.

El autor debe encargarse de realizar las modificaciones necesarias y posteriormente el moderador debe verificar que efectivamente se hayan realizado las correcciones y no se hayan introducido nuevos errores.

En general, se recomienda utilizar esta estrategia cuando la cantidad de anomalías sea mínima.

INTRODUCCIÓN A LAS PRUEBAS DE CÓDIGO

El segundo consiste en no aprobar el producto.

En este caso el autor debe encargarse de realizar las modificaciones necesarias y posteriormente debe realizarse otra inspección del producto corregido.

Como mínimo la inspección debe revisar las áreas del producto modificadas para corregir anomalías identificadas en la inspección anterior, así como efectos secundarios debido a los cambios introducidos.

02

JUSTIFICACIÓN Y BENEFICIOS

DuocUC®

JUSTIFICACIÓN Y BENEFICIOS

Mejora de la calidad del software:

Al detectar y corregir errores y problemas en el código, las pruebas de inspección de código contribuyen a un software más confiable, estable y de alta calidad.

Ahorro de costos:

Identificar y solucionar problemas en etapas tempranas del desarrollo evita que se propaguen y se conviertan en errores costosos de corregir en etapas posteriores.

DuocUC®

JUSTIFICACIÓN Y BENEFICIOS

Cumplimiento de estándares de codificación:

Las pruebas de inspección de código permiten asegurarse de que el código cumpla con los estándares de codificación establecidos, lo que facilita la comprensión y el mantenimiento del código por parte de diferentes desarrolladores.

Mejora de la seguridad:

Al analizar el código en busca de vulnerabilidades, las pruebas de inspección de código ayudan a identificar posibles brechas de seguridad y a aplicar las correcciones necesarias para proteger el software contra posibles ataques.

JUSTIFICACIÓN Y BENEFICIOS

Algunos ejemplos de reglas

ID	Regla	Clasificación	Severidad
RI001	Evitar las líneas de más de 80 caracteres, ya que no son manejadas bien por muchas terminales y herramientas.	Performance	Media
RI002	No debe existir código comentado, es decir, los comentarios sólo deben referirse a implementación u observaciones.	Seguridad	Alta
RI003	Se recomienda una declaración por línea, ya que facilita los comentarios.	Mantenibilidad	Baja
RI004	Inicializar las variables locales donde se declaran. La única razón para no inicializar una variable donde se declara es si el valor inicial depende de algunos cálculos que deben ocurrir.	Mantenibilidad	Alta
RI005	Por principios de encapsulación ninguna variable de instancia o clase ha de ser pública sin una buena razón. Para acceder a variables de instancia o clase se utilizan los denominados métodos set o get.	Seguridad	Alta
RI006	Archivos con más de 2000 líneas son engorrosos y deben ser evitados.	Performance	Media
RI007	Todo código fuente debe tener comentarios en el inicio que indiquen versión, además de una breve descripción del propósito de la clase en el programa.	Mantenibilidad	Baja

JUSTIFICACIÓN Y BENEFICIOS

Cuando una expresión no quepa en una sola línea, debe ser cortada de acuerdo a los siguientes principios:			
RI008	Cortar antes de una coma.	Mantenibilidad	Baja
RI009	Cortar después de un operador.	Mantenibilidad	Baja
RI0010	Preferir cortes de alto nivel (más hacia la derecha de la expresión) que de bajo nivel (más a la izquierda que la expresión.)	Mantenibilidad	Baja
RI0011	Alinear la nueva línea con la primera expresión del nivel anterior.	Mantenibilidad	Baja
RI0012	Si las reglas anteriores llevan a un código confuso o a código que se aplastan contra el margen derecho, reemplazar con 8 espacios en su lugar.	Mantenibilidad	Baja
RI0013	Los comentarios de bloque tienen un asterisco "*" antes de cada línea.	Mantenibilidad	Baja
RI0014	No utilizar datos en duro.	Mantenibilidad	Alta
RI0015	No generar código muerto (código declarado y no utilizado)	Performance	Alta
RI0016	Las llaves de apertura deben aparecer al final de la misma línea de la declaración de la sentencia.	Mantenibilidad	Baja

JUSTIFICACIÓN Y BENEFICIOS

Sentencias if, if-else, if else-if else, for, while, do while, switch, try catch			
La clase de sentencias if-else debe tener la siguiente forma: **Código** **1. if (condition) {** **2. /* statement */** **3. }** **4.** **5. if (condition) {** **6. /* statement */** **7. } else {** **8. /* statement */** **9. }** **10.** **11.if (condition) {** **12. /* statement */** **13.} else if (condition){** **14. /* statement */** **15.} else {** **16. /* statement */** **17.}**	Performance	Media	

Nota: La sentencia if siempre debe usar llaves. con el fin de evitar errores**.**

DuocUC®

JUSTIFICACIÓN Y BENEFICIOS

- Una sentencia for debe tener la siguiente forma:

Código

```
1.for(initialization; condition; update) {
2.  /* statements */
3.}
```

Una sentencia for vacía (una en la que todo el trabajo se hace en las cláusulas de inicialización, condición, y actualización) debe tener la siguiente forma:

Código

```
1.for(initialization; condition; update);
```

Al usar el operador coma en la cláusula de inicialización o actualización de una sentencia for, evitar la complejidad de usar más de tres variables.

Si se necesita, usar sentencias separadas antes de bucle for (para la cláusula de inicialización) o al final del bucle (para la cláusula de actualización).

JUSTIFICACIÓN Y BENEFICIOS

Performance	Estas recomendaciones son para hacer que el código funcione de manera eficiente y eficaz
Seguridad	Estas recomendaciones se enfocan en la protección, de manera tal que no se expongan datos, sistemas o cualquier información sensible en el código fuente.
Mantenibilidad	Se sugieren reglas para ordenar y normar la escritura y estructura del código fuente. Estas normas, facilitan la administración de los sistemas involucrados.

03

HERRAMIENTAS

DuocUC®

HERRAMIENTAS

SonarQube: Es una plataforma de análisis estático de código abierto que proporciona métricas de calidad del código, detección de duplicados, revisiones de seguridad y más. Permite realizar un análisis profundo del código y generar informes detallados.
https://docs.sonarsource.com/

Checkstyle: Es una herramienta de análisis estático de código para Java que verifica si el código cumple con las convenciones de codificación y las reglas de estilo predefinidas. Proporciona recomendaciones para mejorar la calidad del código.
https://checkstyle.sourceforge.io/

HERRAMIENTAS

PMD: Es una herramienta de análisis estático de código que identifica problemas de calidad, estilo y rendimiento en el código fuente. Proporciona una amplia gama de reglas y métricas para evaluar la calidad del código.
https://pmd.github.io/

FindBugs: Es una herramienta de análisis estático de código para Java que encuentra posibles errores de programación en el código fuente. Identifica patrones de código sospechoso que podrían conducir a errores o vulnerabilidades.
https://findbugs.sourceforge.net/

HERRAMIENTAS

ESLint:

Es una herramienta de análisis estático de código para JavaScript que ayuda a encontrar problemas de calidad y estilo en el código.

Detecta errores comunes, posibles problemas de rendimiento y sugerencias de mejoras de código.
https://eslint.org/

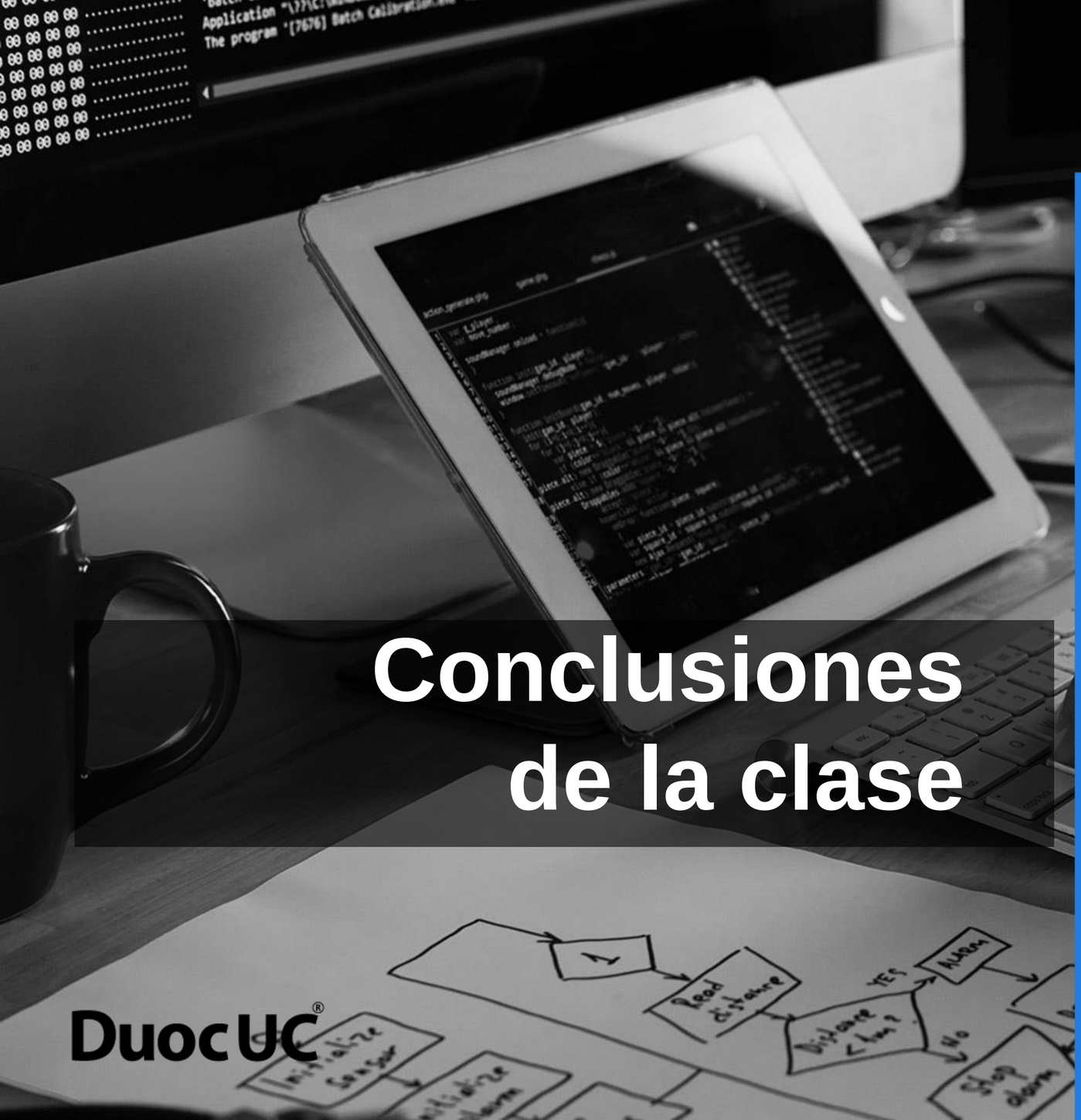
HERRAMIENTAS

Estas herramientas de inspección de código brindan funcionalidades avanzadas de análisis estático, generación de informes detallados y recomendaciones específicas para mejorar la calidad del código.

Se pueden integrar en el flujo de desarrollo para realizar pruebas automáticas y brindar retroalimentación en tiempo real a los desarrolladores, lo que ayuda a garantizar un código más limpio, eficiente y confiable.

DuocUC®

Conclusiones de la clase



DuocUC®

Conclusiones

- ✓ Las pruebas de inspección de código son un proceso de revisión sistemática y exhaustiva del código fuente de un programa para identificar errores, vulnerabilidades, malas prácticas y oportunidades de mejora.

- ✓ Al detectar y corregir errores y problemas en el código, las pruebas de inspección de código contribuyen a un software más confiable, estable y de alta calidad.

Bibliografía

- ✓ Verity. (6 de Septiembre 2022). ¿Qué es la inspección de código fuente?. https://www.verity.cl Recuperado de https://www.verity.cl/code-review/

DuocUC®

DuocUC® INFORMÁTICA Y TELECOMUNICACIONES

Importancia y técnicas de inspección de código

Calidad de Software - CSY4111