

# Análisis de beneficios y riesgos de usar la cobertura de código

¿**Qué** es un caso de prueba?

¿**Qué** es la cobertura de código?

¿**Para qué** sirve la cobertura de código?

# Análisis de beneficios y riesgos de usar la cobertura de código

*Es una especificación de una entrada (inválida o válida) como argumento a una función de un programa informático.*

Pág. 2

# Análisis de beneficios y riesgos de usar la cobertura de código

*Es el proceso por el cual “mediante métricas” se genera un reporte que identifica código que ~han ejecutado una funcionalidad.*

Pág. 2

# Análisis de beneficios y riesgos de usar la cobertura de código

*Evalúan la efectividad de los casos de prueba y determinan el comportamiento del programa informático ante una entrada ~válida.*

Pág. 2

# Análisis de beneficios y riesgos de usar la cobertura de código

*Refactorización de código, identificación de código  
con altas frecuencias de ejecución y depuración  
relacionada con el rendimiento.*

Pág. 2

# Planteamiento del Problema

*“La revisión eficaz de un programa es imprescindible para cualquier programa informático complejo”.*

Miller y Maloney (1963)

# Planteamiento del Problema

*Fallos ocasionados en los programas informáticos eran causados por condiciones entrantes que derivaron en la ejecución de código que se ejecutaba en condiciones poco probables.*

Miller y Maloney (1963)

# Planteamiento del Problema

*¿Cómo confiar en un programa informático de cual no se tiene certeza si funcionara correctamente ante la entrada de condiciones anormales?*



# Planteamiento del Problema

La verdadera pregunta:

*¿Se puede confiar en un programa informático para que cumpla sus especificaciones funcionales con cada nueva condición de entrada anormal?*

# Justificación

La cobertura de código, se centra en el aspecto estructural del código.

(Su ejecución)

Evaluación de los casos de prueba.

# Estado del Arte

Es aplicable en la etapa de pruebas, casos unitarios (Unit Testing), pruebas de integración (Integration Testing) y pruebas del sistema (System Testing).

# Evaluación de casos de prueba

Motivos cobertura de código:

- Incremento fiabilidad (elimina casos redundantes).
- Proporcionar medida cuantificable del progreso.
- Priorización de pruebas

# Herramientas de Cobertura de Código

Dos aproximaciones conceptuales.

- Instrumentación por sobrecarga fuera de línea.
- Instrumentación por sobrecarga.

# Nuevas formas de usar la cobertura de código

- Identificación código alta frecuencia de ejecución.
- Refactorización de código.
- Depuración relacionada con el rendimiento.

# Metodología

*Metodología CDIO (concebir, diseñar, implementar y operar), comprender importancia y impacto investigación y desarrollo tecnológico, aumento productividad ambientes complejos basados tecnología y procesos tecnológicos.*

# El diseño conceptual

Maximilian, se observó módulos donde reside código que concentra la mayoría del tiempo computacional y que carga con altas tasas de ejecución.



# El diseño conceptual

Pugi se evaluó los casos de prueba y se determino que tan bien cubren el código.

Metodología TDD

# Análisis de Medidas

La cobertura de código no dará  
una evaluación a aquella  
funcionalidad no implementada.

# Análisis de Medidas

Programas heredados, diseño y ejecución casos de prueba cuando son inexistentes.

# Análisis de Medidas

Observar una regresión través  
de ejemplos toman papel caso  
prueba.

# Análisis de Medidas

Maximilian determinar bloques de código redundantes o con altas tasas de ejecución.

# Filtrado de Métricas

Los reportes filtran y excluyen condiciones ilegales, lógica sin usar o baja prioridad, áreas de código funcionamiento verificado.

# Filtrado de Métricas

Aquellas condiciones ilegales son clausulas "if" y "else" que evalúan casos sin uso o segmentos de código predeterminados que no deben de suceder bajo ciertas condiciones de entrada, como la ejecución de código específico a una plataforma o biblioteca.

# Implementación

- Herramienta instrumentación gcov v9.2 (Nov. 2019)  
*(herramienta de profiling, baja sobrecarga de ejecución  
(acoplamiento con gprof), integración con IDE).*
- Compilador gcc v9.2 (Nov. 2019).



# Implementación

Soporte a C++17, banderas optimización deshabilitadas, .

Necesarias para obtener resultados congruentes.

# Implementación

Soporte a C++17, banderas de cobertura habilitadas,

Requisito herramientas  
cobertura de código.

# Resultados y Discusión

Para Maximilian, se identifico bloques de código con altas tasas de ejecución son los encargados de conversión del buffer de valores.

# Resultados y Discusión

Para Maximilian, se determino código redundante que logró ser simplificado mediante refactorización.

# Resultados y Discusión

Para Pugi, la metodología de desarrollo es TDD, se usó desde el la etapas iniciales de la implementación y los reportes de cobertura de código muestran que.

# Resultados y Discusión

**946**  
**Casos de**  
**Prueba**

**100%**  
**Archivos**

**99%**  
**Lineas de**  
**Código**

# Resultados y Discusión

Confirmado las conclusiones de Derezińska (2008), quien afirma el alto porcentaje de cobertura de código que se debe de lograr usando metodologías como TDD.

# Conclusiones

Encargado de los casos pruebas, la cobertura de código asegura que los planes de verificación y pruebas trazadas para módulos y funciones cubre la totalidad del código.



# Conclusiones

Para programadores, la cobertura de código asegura que la totalidad de líneas escritas para un programa informático han sido probadas.

# Conclusiones

Para equipos de desarrollo centrados en las pruebas de regresión, la cobertura de código simplifica su labor, porque identifica código no ejecutado por la casos de prueba ya existentes.

# Conclusiones

**Beneficios directos de la cobertura de código, determinar la eficacia de los casos de prueba.**

# Conclusiones

**Identificar zonas de código  
con altas frecuencias de  
ejecución.**

# Conclusiones

**Identificar bloques de  
código con nula  
funcionalidad**

# Conclusiones

**Depuración de programas  
con problemas de  
rendimiento**

# Conclusiones

**Refactorización código**

# Conclusiones

Otras propuestas, Véase. Tikir y  
Hollingsworth (2002)



# Conclusiones

Utilizar la cobertura de código como herramienta de retroalimentación.

# Conclusiones

**Identificar funcionalidad  
pobremente usada por los  
usuarios y patrones de uso  
inesperados.**

# Conclusiones

**Priorizar zonas de código,  
donde la cobertura de  
código es baja.**

# Conclusiones

**Medir el progreso realizado  
en las pruebas.**

# Conclusiones

Metodologías como TDD permiten una alta cobertura de código desde el inicio y su facilidad permite que principiantes logren una alta tasa de cobertura de código con poco esfuerzo.

# Conclusiones

Riesgo de la cobertura de código, desenfoque de áreas prioritarias, diseño, preparación e implementación de nueva funcionalidad.

¿Contradicción?

# Conclusiones

La implementación de casos de prueba es una tarea que necesita del dominio del área específica y los conocimientos necesarios para decir que un programa informático está bien probado.

# Conclusiones

Implementar casos de prueba para aumentar la cobertura de código no garantizará la ausencia de errores.



# Conclusiones

No existe una teoría subyacente que determine qué tanto mejora la eficacia de los casos de prueba con una cobertura de código alta.

# Conclusiones

Problemas de interpretación en las métricas, confunden, obteniendo resultados dispares entre diferentes métricas a un mismo caso de prueba ejecutado.

# Conclusiones

Métricas de cobertura de código afectadas negativamente en fiabilidad entre más grande es el programa informático.