PRUEBAS DE MUTACIÓN EN GOOGLE

IYÁN SOLÍS RODRÍGUEZ | UO295103

DAVID ÁLVAREZ CABEZAS | UO293944

Información de la entrevista:

ENTREVISTADO: GORAN PETROVIC

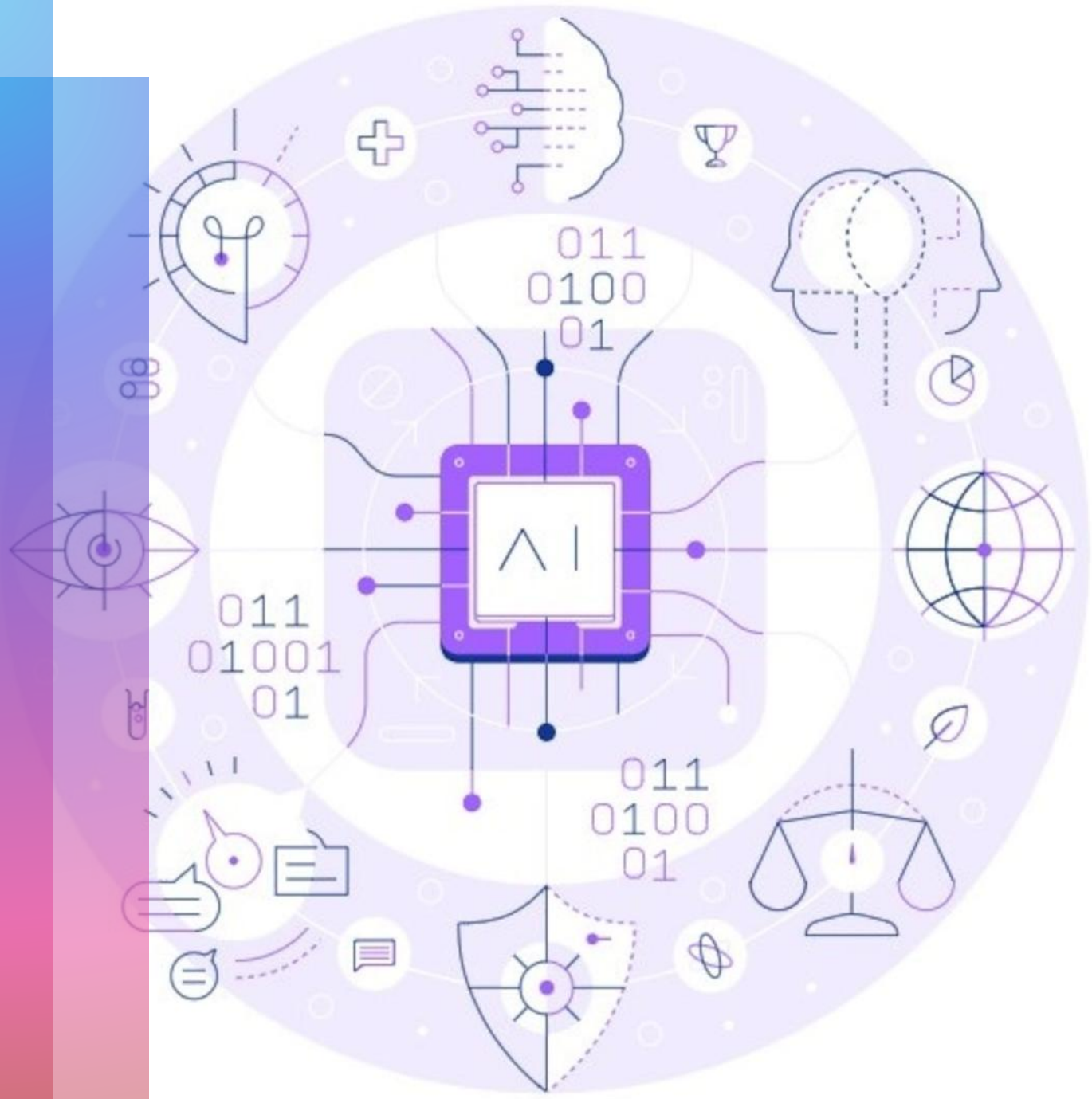
TRABAJADOR DE GOOGLE

¿QUÉ SON LAS PRUEBAS DE MUTACIÓN?

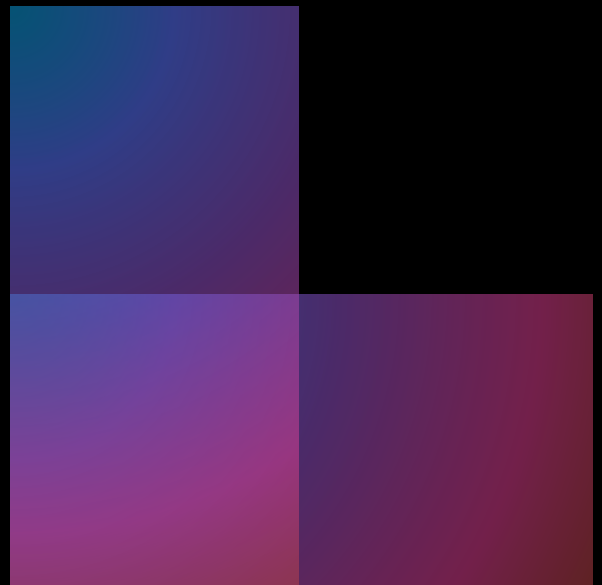
- Las pruebas de mutación se pueden definir como una evaluación de la calidad del software mediante inserción de pequeños errores y medición de detección.
- Al inyectar mutaciones, podemos descubrir qué partes del código podrían ser vulnerables a fallos y mejorar la robustez del software en general.
- Esta metodología ha ganado popularidad en empresas como Google porque permite a los desarrolladores detectar problemas antes de que afecten a los usuarios finales.

¿POR QUÉ Y CÓMO FUNCIONAN?

- Cobertura de líneas no es suficiente.
- Validación de pruebas efectivas.
- Creación de versiones mutadas del código (mutantes).
- Ejecución de pruebas y cálculo del puntaje de mutación.



Ejemplos y Beneficios



Ejemplos: cambio de operadores aritméticos, eliminación de líneas, modificación de condiciones lógicas...

Mejora la calidad del software.

Ayuda en la refactorización del código.

Identificación de pruebas inefectivas.

Efecto de Acoplamiento

- El efecto acoplamiento parte de la premisa de que los mutantes generados no se parecen a errores humanos, pero la hipótesis de acoplamiento trata de como estos son similares a los errores humanos de tal manera que los caos de prueba que detecten estos mutantes también lo harán con errores humanos.

Implementación en Google

- Crecimiento rápido: de 93 revisiones en 2016 a miles de usuarios hoy.
- Herramienta Mutagénesis integrada en revisión de código.
- Generación y ejecución automatizada de mutantes.

Distintos lenguajes de programación

¿SE TRATAN DE LA MISMA MANERA EN TODOS LOS LENGUAJES DE PROGRAMACIÓN?

¿LAS HERRAMIENTAS ESTÁN HECHAS EN EL PROPIO LENGUAJE?

¿POR QUÉ NO TRATARLOS POR IGUAL?

- C++
- Python
- Java
- Go
- JavaScript
- TypeScript
- Kotlin
- Dart
- Common Lisp
- SQL
- Rust

¿Cuándo se ejecutan las pruebas?

- Integradas en la revisión automática de código
- Tras hacer una pull request
- Una herramienta más entre los cientos de analizadores que usamos constantemente

¿Cómo se ejecutan?

- Problemas:
  - Gran cantidad de pruebas realizadas (500 millones)
  - Alto coste computacional
  - ¿Cómo hacemos esto escalable?

- Soluciones:
  - Optimización
  - Heurística de supresión agresiva
  - Alta capacidad de caché de Google
  - Metamutante

Consecuencias de su implementación

- Sobrecarga al desarrollador
- Mayor número de pruebas
- Mayor calidad de las pruebas implementadas
- Eliminación de pruebas "sin valor"
- En general, mejor software

Experiencia personal de Goran

- Heredado "obligado" de un hackathon
- Convertido en proyecto personal
- Aplicación de un desarrollo basado en capacidad para hacerlo viable
- Desarrollo constante de nuevos heulísticos
- Experiencias personales con su uso
- Consejos para introducirlo en las empresas

¿PREGUNTAS?