FACITEC
Facultad de Ciencias y Tecnología
Universidad Nacional de Canindeyú

WORKSHOP
INVESTIGACIÓN, INNOVACIÓN Y EXTENSIÓN
14 DE OCTUBRE DE 2022

EXPOTEC
5° Edición 2022

Prueba de implementación de *Random Testing* en *JavaScript*

Autores:
Nazario Luis Ayala Frasnelli; Fátima Aidee Cáceres Urdapilleta y Carlos Cesar Golin Galeano
Correo electrónico:
nazarioayala@facitec.edu.py; fatimacaceres@facitec.edu.py; carlosgolin@unican.edu.py

Palabras clave: testing, random testing, calidad de software

INTRODUCCION

Es fundamental que los equipos de desarrollo cuenten con estrategias para detectar y solucionar errores, y de este modo evitar la presencia de defectos en los productos de *software* entregados a los clientes.

En el presente trabajo se aborda una de las técnicas de pruebas de *software*, la cual se denomina *Random Testing*.

OBJETIVO

El objetivo del trabajo es realizar una implementación práctica de la técnica *Random Testing* en el lenguaje Javascript.

METODOLOGIA

- Desarrollo de librería.
- Desarrollo de caso de prueba

RESULTADOS

RandomizedTestingTool.js

```
module.exports = class {
    //Metododo de ejecución de pruebas aleatorias
    static run_test(test_function, expected_function, generator, iteration = 100){
        var errors = []
        for (let i = 0; i < iteration; i++) {
            //Generación de entrada aleatoria
            var random_input = generator()
            //Ejecución de función a ser probada
            var test_result = test_function(random_input)
            //Ejecución de fucnión de validación
            var expected_result = expected_function(random_input)

            //Verificación y registro de ocurrencia de error
            if(test_result != expected_result)
                errors.push({
                    input:random_input,
                    got: test_result,
                    expected: expected_result,
                })
        }
        return errors
    }
}
```

Fig. 1 Herramienta de ejecución de *Random Testing*

absTest.js

```
//Impertación de clase a ser probada
var NumberUtil = require("./NumberUtils")
//Importación de la herramienta de ejecución de pruebas aleatorias
var RandomizedTestingTool = require("./RandomizedTestingTool")

//Generador pseudoaleatorio de entradadas
var integerGenerator = function (min = -10, max = 10) {
    return Math.floor(Math.random() * (max - min + 1) + min)
}

//Ejecución de la prueba
var detected_errors = RandomizedTestingTool.run_test(
    NumberUtil.abs,     //función a probar
    Math.abs,           //función que establece el resultado esper
    integerGenerator    //Generador de entradas
)

//Impresión de errores encontrados
console.log(detected errors);
```

NumberUtils.js

```
module.exports = class {
    //Método que calcula el valor absoluto
    static abs(x){
        //Error introducido intencionalmente
        return (x >= 0) ? x : x;
    }
}
```

```
{ input: -10, got: -10, expected: 10 },
{ input: -5, got: -5, expected: 5 },
{ input: -5, got: -5, expected: 5 },
{ input: -8, got: -8, expected: 8 },
{ input: -6, got: -6, expected: 6 },
{ input: -10, got: -10, expected: 10 },
{ input: -2, got: -2, expected: 2 },
{ input: -1, got: -1, expected: 1 },
{ input: -6, got: -6, expected: 6 },
{ input: -1, got: -1, expected: 1 },
```

Fig. 2 Ejecución del caso de prueba

CONCLUSION

Las pruebas realizadas en el trabajo son apenas una demostración de lo que puede realizarse aplicando esta técnica y se requiere de un análisis de la misma ante escenarios más complejos.

BIBLIOGRAFIA

[1] T. R. Devi, “Importance of Testing in Software Development Life Cycle,” Int. J. Sci. Eng. Res., vol. 3, no. 5, 2012.

[2] T. Y. Chen, F.-C. Kuo, D. Towey, and Z. Q. Zhou, “A revisit of three studies related to random testing,” Science China Information Sciences, vol. 58, no. 5. pp. 1–9, 2015.