Pruebas basadas en mutación

Pedro Reales Mateo[1],

[1] Departamento de Tecnologías y Sistemas de la información,
Universidad de Castilla-La Mancha, Ciudad Real, España
pedro.reales@uclm.com

Resumen. Las pruebas basadas en mutación han sido tradicionalmente usadas en la investigación para evaluar métodos de generación de pruebas. Sin embargo, actualmente estas técnicas son suficientemente maduras para transferir estas prácticas a la industria. El tutorial propuesto en este documento pretende dar a conocer estas técnicas desde un punto de vista práctico, para que tanto investigadores como desarrolladores sean capaces de usar técnicas de mutación para evaluar sus pruebas y métodos de pruebas.

Palabras clave: Pruebas, mutación, costes, herramientas de mutación y proceso de mutación.

1 Introducción

Una de las técnicas de pruebas más efectivas son las pruebas basadas en mutación [1]. Con esta técnicas se consiguen realizar pruebas exhaustivas de un sistema de manera metódica y pudiendo obtener resultados muy buenos.

En el lado izquierdo de la

Tabla 1 se ofrece un pequeñísimo ejemplo de un programa que se desea probar (programa *P*) y que consiste en una sencilla función Java que devuelve la suma de los dos números que se le pasan como parámetro. Debajo aparecen cuatro mutantes, cada uno de los cuales contiene una modificación sintáctica: en los mutantes 1, 2 y 3, el operador + se ha sustituido por, respectivamente, los operadores -, * y /; en el mutante 4 se ha respetado el operador aritmético, pero se ha añadido a la variable *b* un operador de postincremento (++).

En el lado derecho aparecen cuatro posibles casos de prueba para este sistema, y se muestran los resultados que devuelve cada uno de los cinco programas con los datos de prueba: al ejecutar *P* con los parámetros 1 y 1, el resultado devuelto es 2; no obstante, al ejecutar el *Mutante 1* con los mismos valores, el resultado es 0, ya que el + se ha sustituido por un -. El *Mutante 2* devuelve también 1, porque lo que realmente hace es multiplicar los dos valores; el *Mutante 3* los divide, devolviendo también 1. El *Mutante 4*, sin embargo, devuelve 2 (igual que *P*) ya que el valor de *b* se incrementa después de devolver el resultado, por lo que éste no es observable desde el exterior. De este modo, el caso de prueba *(1, 1)* mata a los mutantes 1, 2 y 3; el caso (0, 0)

mata solamente al *Mutante 3*; *(-1, 0)* mata a los mutantes 2 y 3; por último, *(-1, -1)* mata a los mutantes 1, 2 y 3 ya que devuelven resultados distintos del devuelto por el programa original.

El *Mutante 4*, sin embargo, permanece vivo con estos cuatro casos de prueba y, además, permanecerá siempre vivo y será imposible de matar, se le pase el caso de prueba que se le pase. A estos mutantes cuyo comportamiento es siempre exactamente igual al del programa original se los llama *mutantes funcionalmente equivalentes* o, simplemente, *mutantes equivalentes,* y realmente representan "ruido" que dificulta el análisis de los resultados (es decir, conocer el porcentaje real de mutantes que mata el *test suite*).

Tabla 1. Un programa original (P), cuatro mutantes y los resultados de cada mutante con algunos casos de prueba.

Versión	**Código**

P (original)	int sum(int a, int b) {
	return a + b;
	}

Mutante 1	int sum(int a, int b) {
	return a **-** b;
	}

Mutante 2	int sum(int a, int b) {
	return a * b;
	}

Mutante 3	int sum(int a, int b) {
	return a **/** b;
	}

Mutante 4	int sum(int a, int b) {
	return a + b**++**;
	}

	Datos de prueba			
	(1, 1)	(0, 0)	(-1, 0)	(-1, -1)
P	2	0	-1	-2
Mutante 1	0	0	-1	0
Mutante 2	1	0	0	1
Mutante 3	1	Error	Error	1
Mutante 4	2	0	-1	-2

Para medir la calidad del *test suite*, por tanto, se necesita conocer el número de mutantes generados, el número de mutantes muertos y el número de mutantes equivalentes. La calidad se mide con el *mutation score*, que viene dado por la expresión de la **Fig 1**.

$$MS(P,T) = K / (M-E)$$, donde:

P : programa bajo prueba
T : *test suite*
K : número de mutantes muertos
M : número de mutantes generados
E : número de mutantes equivalentes

Fig 1. Cálculo del mutation score

Las técnica de mutación resulta de especial interés en el ámbito de la investigación, pues aportan a los investigadores un método cuantitativo muy efectivo para evaluar la calidad de las pruebas generadas o diseñadas a partir de los resultados de sus investigaciones.

Así mismo, en el ámbito industrial, y gracias al gran avance en los últimos años en la reducción de los costes de la mutación (*high order mutation* [2], *mutant schema* [3], *flexible weak mutation* [4], selective mutation [5], etc…), este tipo de pruebas puede ser especialmente útil cuando haya que realizar pruebas exhaustivas de algún componente de un sistema.

Desafortunadamente, las técnicas de pruebas basadas en mutación no son muy conocidas en la comunidad científica y prácticamente desconocidas en la industria [6].

2 Objetivos

Este tutorial pretende dar a conocer las técnicas de pruebas basadas en mutación a un nivel teórico y práctico, dando una visión útil tanto para investigadores en el área de las pruebas como a desarrolladores y testers de software.

El objetivo final del tutorial es que los asistentes obtengan una visión general de la mutación como una técnica para evaluar conjuntos de casos de prueba, que sean capaces de diseñar nuevas pruebas de calidad y de aplicar estas técnicas tanto a nivel de investigación (para evaluar métodos de generación o diseño de pruebas), como a nivel profesional (para diseñar y evaluar de manera cuantitativa y objetiva las pruebas de un sistema).

3 Resumen

Para cumplir los objetivos del tutorial, se expondrán en primer lugar los aspectos más relevantes de la mutación: conceptos básicos, proceso de mutación, técnicas de reducción de costes, herramientas, etc…

Después de la sesión teórica se realizará una sesión práctica usando la herramienta de mutación Bacterio (http://www.alarcosqualitycenter.com/index.php/productos/bacterio), desarrollada y proporcionada por el grupo de investigación Alarcos. Durante esta sesión se pondrá en práctica el proceso de mutación y se explorarán todas las técnicas de mutación disponibles e implementadas en la herramienta (mutación unitaria y a nivel de sistema [4]; mutación fuerte [1], débil [7], flexible [4], *functional qualification* [8]; mutación de orden *n* [2]; ejecución en paralelo [9]; mutación selectiva [10]; mutación aleatoria [11]).

4 Audiencia potencial

El tutorial está dirigido a profesores de ingeniería del software, alumnos de informática, investigadores en el ámbito de las pruebas, desarrolladores de sistemas y profesionales dedicados a hacer pruebas de software.

5 Ponente

Pedro Reales Mateo, Grupo de investigación Alarcos, Universidad de Castilla-La Mancha.

Pedro Reales realiza su tesis doctoral en el ámbito de las pruebas del software en la Universidad de Castilla-La Mancha. Es ingeniero en Informática (perfil de Sistemas de Información Universidad de Castilla-La Mancha, 2008). Sus líneas de investigación están relacionadas con la generación automática de casos de prueba, pruebas para líneas de producto software y pruebas basadas en mutación.

6 Esquema de contenidos

Parte teórica. ***45 minutos***

Pruebas basadas en mutación.
Durante esta parte se explicarán los conceptos teóricos básicos de las pruebas basadas en mutación, el proceso de mutación con dos variantes (una para investigadores y otra para desarrolladores) y se mostrarán las ventajas y desventajas de este tipo de pruebas.

Técnicas de mutación.
Se explicarán las técnicas de mutación más importantes, dando ideas de cuándo deben ser usadas. El conocimiento de esta técnicas en básico para aplicar las pruebas de mutación correctamente.

Parte práctica. ***45 minutos***

Introducción a Bacterio Mutation System.
Aquí se presentará la herramienta Bacterio, se describirán todas las características y técnicas que implementa, se hará una breve descripción del proceso de instalación y configuración y se mostrará un breve ejemplo de cómo se realiza el proceso de mutación.

Ejemplo práctico del proceso de mutación con Bacterio. 1hora
En esta parte se pretende realizar un proceso completo de mutación, simulando el proceso que seguiría un desarrollador o un tester. El objetivo de esta parte es que los asistentes usen Bacterio y lleven a cabo un proceso de mutación completo.

7 Medios necesarios para el desarrollo del tutorial

Para la parte teórica serán necesarios los elementos típicos para realizar una presentación: un proyector y un ordenador (aportado por el ponente). Así mismo, sería de utilidad repartir copias de las presentaciones a los asistentes.

Para la parte práctica serán necesarios un ordenador para cada uno de los asistentes, aunque pueden usar sus propios portátiles. El ponente proporcionará una copia del software necesario para seguir el tutorial y copias de la presentación para cada asistente.

8 Resultados previos

Como se ha mencionado, el ponente está realizando su tesis doctoral en el ámbito de las pruebas de software, poniendo especial énfasis en la mutación. Como parte de su tesis, ha desarrollado la herramienta Bacterio, que se utilizará en la segunda parte del tutorial.

A este respecto, se han publicado los siguientes resultados:

1. "Mutation Testing Cost Reduction Techniques: A Survey". Este artículo fue publicado en la revista IEEE Software en Mayo de 2010. En él se describen una revisión sobre diferentes técnicas para reducir los costes de la mutación, así como una guía de qué deben implementar las herramientas de mutación para que sean efectivas. La mayoría de las técnicas de mutación presentadas en este artículo han sido implementadas en la herramienta Bacterio.
2. "Mutation at System and Functional Levels". Este artículo fue publicado en la conferencia MUTATION'10 en Abril del 2010. En este artículo se expone una novedosa técnica de mutación especialmente diseñada para trabajar con sistemas multiclase y sistemas completos llamada "Flexible Weak Mutation". Este artículo fue seleccionado para extenderlo y publicarlo en un special issue sobre mutación de la revista Science of Computer Programming. La herramienta Bacterio también implementa la técnica de mutación "Flexible Weak Mutation".
3. "Mutation at the Multi-Class and System levels". Artículo extendido del anterior, que ha sido recientemente aceptado, pero todavía no publicado. Además de lo comentado anteriormente, este artículo presenta una serie de nuevos operadores de mutación diseñados para simular errores en sistemas completos. Estos operadores están actualmente bajo revisión y serán implementados en la herramienta Bacterio cuando se vayan a evaluar empíricamente.

Finalmente destacar que el ponente realizó una estancia de seis meses en la universidad George Mason University, Fairfax, UU.EE. bajo la supervisión del profesor Jeff Offutt, uno de los investigadores más activos en el ámbito de las pruebas basadas en mutación.

Referencias

[1] DeMillo, R., Lipton, R.J., and Sayward, F.G., *Hints on test data selection: Help for the practicing programmer*. IEEE computer, 1978. **11**(4): p. 34-41.

[2] Polo, M., Piattini, M., and García-Rodríguez, I., *Decreasing the cost of mutation testing with second-order mutants*. Software Testing, Verification and Reliability, 2008. **19**(2): p. 111-131.

[3] Untch, R., Offutt, A., and Harrold, M. *Mutation analysis using program schemata*. In *International Symposium on Software Testing, and Analysis*. June 28-30, 1993. Cambridge, Massachusetts: ACM Press.

[4] Reales, P., Polo, M., and Offutt, J. *Mutation at System and Functional Levels*. In *Third International Conference on Software Testing, Verification, and Validation Workshops*. April, 2010. Paris, France.

[5] Barbosa, E.F., Maldonado, J.C., and Auri Marcelo Rizzo Vincenzi, *Toward the determination of sufficient mutant operators for C*. Software Testing, Verification and Reliability, 2001. **11**(2): p. 113-136.

[6] Polo, M. and Reales, P., *Mutation Testing Cost Redution Techniques: A Survey*. IEEE Software, 2010. **27**(3): p. 80-86.

[7] Offutt, A.J. and Lee, S.D., *An Empirical Evaluation of Weak Mutation*. IEEE Transactions on Software Engineering, 1994. **20**(5): p. 337-344.

[8] Bombieri, N., Fummi, F., Pravadelli, G., Hampton, M., and Letombe, F. *Functional qualification of TLM verification*. In *Design, Automation and Test in Europe, DATE'09*. April 20-24, 2009. Nice, France.

[9] Offutt, A.J., Pargas, R.P., Fichter, S.V., and Khambekar, P.K. *Mutation Testing of Software Using a MIMD Computer*. In *International Conference on Parallel Processing*.

[10] Offutt, A.J., Rothermel, G., Untch, R.H., and Zapf, C., *An experimental determination of sufficient mutant operators*. ACM Transactions on Software Engineering and Methodology, 1996. **5**(2): p. 99-118.

[11] King, K.N. and Offutt, A.J., *A Fortran language system for mutation based software testing*. Software: Practice and Experience, 1991. **21**(7): p. 685-718.