Revisión de Resultados Experimentales sobre Performance de Técnicas Pruebas de Software

Mario L. Guerini
Centro de Ingeniería de Software e Ingeniería del Conocimiento. Escuela de Postgrado. ITBA.
Argentina

Enrique Fernández, Alejandra Ochoa, Hernán Merlino
Laboratorio de Sistemas Inteligentes. Facultad de Ingeniería. Universidad de Buenos Aires.
Argentina
Centro de Ingeniería de Software e Ingeniería del Conocimiento. Escuela de Postgrado. ITBA.
Argentina

Eduardo Diez, Paola Britos y Ramón García-Martínez
Centro de Ingeniería de Software e Ingeniería del Conocimiento. Escuela de Postgrado. ITBA.
Argentina
Laboratorio de Sistemas Inteligentes. Facultad de Ingeniería. Universidad de Buenos Aires.
Argentina
rgm@itba.edu.ar

Abstract

This paper describes experimental results revalidation in software testing techniques using statistical non-parametric test. The obtained results show that meanwhile the isolated analysis of the experimental results shows that mutation based techniques are more effective than all-uses technique for failure detection, the global parameters evaluation shows that there is no evidence to support this difference..

Keywords: Software experiment revision. Software proof techniques.

Resumen

En el presente trabajo se describe el uso de un test no paramétrico para revalidar los resultados experimentales previos sobre el desempeño de cuatro técnicas de prueba de software. El mismo permite demostrar que, si bien el análisis aislado de los distintos experimentos y la evaluación de los valores a nivel global parecieran indicar que las variantes de la técnica de mutación son más efectivas en la detección de fallas que la técnica de all-uses, no existe evidencia empírica que permita afirmar esta hipótesis.

Palabras claves: Revisión de experimentos de software. Técnicas de prueba de software.

1. INTRODUCCION

1.1. Las Pruebas de Software

Bajo el nombre de pruebas de software se agrupan un conjunto de prácticas correctivas (frente a las prácticas preventivas que se aplican durante el proceso de construcción de software) cuyo objetivo es determinar la calidad de los sistemas software [1].

Las técnicas de pruebas de software se pueden agrupar en las siguientes categorías:

- Técnicas Aleatorias: Los casos se prueba se generan aleatoriamente.
- Técnicas Funcionales: Se utilizan las especificaciones del problema para generar los casos de prueba. El programa se ve como una caja negra.
- Técnicas de Flujo de Control: Se requiere conocimiento del código fuente. Se seleccionan caminos dentro del programa que deben ser ejecutados al ingresar los casos de prueba.
- Técnicas de Flujo de Datos: También requiere conocimiento del código fuente. En este caso, los caminos se eligen de forma de explorar secuencias de eventos relacionadas con el estado de las variables.
- Técnicas de Mutación: En la mutación se introducen fallas al programa creando varios mutantes, cada uno con una falla. Los casos de prueba se hacen pasar por los mutantes con la intención de hacer fallar al programa. Cuando ocurre la falla de un mutante, se dice que éste ha sido matado y no se prueba más ese mutante muerto. Asimismo, el caso de prueba que hizo fallar al mutante, se marca como un caso de prueba útil para la detección de fallas. Al cabo de ejecutar todos los casos de prueba, los mutantes pueden sobrevivir debido a dos razones: una es que sean equivalentes al programa inicial, y otra es que no haya habido un caso de prueba lo suficientemente bueno como para detectar el fallo. En este caso, se deben generar más casos de prueba para poder matar todos los mutantes no equivalentes. El score de mutación es el porcentaje de mutantes no equivalentes que mata un conjunto de prueba, cuyo valor ideal es el 100%.

Para una descripción más detallada de las técnicas se puede consultar [2]; [3]. Los estudios sobre estas técnicas se pueden realizar de dos formas: una es intra familias y la otra es entre familias. En el primer caso se comparan técnicas pertenecientes a la misma categoría, mientras que en el segundo, la comparación se hace entre categorías distintas. Las comparaciones hechas en este artículo serán de ambos tipos ya que se compararán tres técnicas de mutación y una de flujo de datos entre sí. En las siguientes subsecciones se describen las técnicas de prueba sobre las que se realizarán los experimentos:

1.2. Técnica de Mutación

Las técnicas de mutación se basan en la modelización de las faltas típicas que se comente al hacer un programa, mediante lo que se conoce como operadores de mutación (dependientes del lenguaje de programación). Cada operador de mutación se aplica sobre el programa, dando lugar a una serie de mutantes (programas exactamente igual al base, pero con una sentencia modificada, precisamente la originada por el operador de mutación). Una vez que se tiene generado el conjunto

de mutantes, se generan casos de prueba que ejerciten la parte mutada del mismo. Tras generar casos de prueba para cubrir todos los mutantes, teóricamente se tienen cubiertas todas las posibles faltas cometidas (en la práctica, sólo las faltas modelizadas por los operadores de mutación).
En este artículo se examinarán los tres criterios de generación de casos de pruebas descriptos en la siguiente tabla:

TÉCNICA	CRITERIO DE GENERACIÓN DE MUTANTES
Mutación (Standard)	Se seleccionaron todos los operadores usados por Monthra [4] excepto el operador GOTO debido a que no era utilizado en los programas de prueba escritos en el lenguaje C.
Mutación abs/ror	Solamente utiliza los operadores abs y ror para generar los mutantes. El operador abs reemplaza el valor de cada variable *x* por abs(*x*), -abs(*x*) y zpush(*x*). El operador zpush hace que el mutante muera inmediatamente si su argumento es cero, lo que requiere que los datos de prueba fuercen a que toda expresión adquiera el valor cero. Cuando se aplica a un programa que contiene una asignación z := x+1, el operador abs genera seis mutantes, obtenidos de reemplazar la asignación por z := abs(x)+ 1, z := -abs(x)+ 1, z := zpush(x)+ 1, z := abs(x + 1), z := -abs(x + 1), y z := zpush(x + 1). El operador ror genera mutantes reemplazando cada operador relacional por otros operadores relacionales. Por ejemplo, cuando se aplica a un programa que contiene un predicado “if (x = 0) then”, el operador genera las siguientes siete condiciones: “if (x < 0) then”, “if (x ≤ 0) then”, “if (x ≠ 0) then”, “if (x > 0) then”, “if (x ≥ 0) then”, “if (true) then”, y “if (false) then”
Mutación 10%	En este caso, se seleccionan aleatoriamente el diez por ciento de los mutantantes generados para cada tipo de mutación.

Tabla 1. Técnicas de mutación

1.3. Técnica de Flujo de Datos

Sean s_i y s_j; $1 \leq i$; $j \leq n$ dos sentencias en un programa (P) donde la variable x es definida y usada respectivamente. Nos referimos a esta definición como $d_i(x)$, y al uso como $u_j(x)$. El par $(d_i(x); u_j(x))$ es llamado *par du*. Un *par du* puede ser *p-use* o *c-use* dependiendo de si s_j es un predicado o no, respectivamente. Este par es *factible* si existe un caso de prueba t en el dominio (D) de todos los casos de pruebas posibles, tal que la ejecución de P en t cause que el control del programa vaya de s_i a s_j, pasando por una o más sentencias que no definan a x. Un camino de este tipo es llamado un *camino libre de definición* con respecto a x. Un caso de prueba t perteneciente a un conjunto de pruebas T cubre un par c-use $(d_i(x); u_j(x))$ si la ejecución de P en t causa la ejecución de un camino libre de definición con respecto a x de s_i a s_j. Un par p-use $(d_i(x); u_j(x))$ es considerado cubierto si un camino libre de definición con respecto a x desde s_i a s_j a s_k es ejecutado por cada sucesor s_k de s_j. Nos referimos al conjunto de todos los pares c-use y p-use como *all-uses*. Un conjunto de prueba T puede ser evaluado contra el criterio *all-uses* computando la razón entre el número total de *all-uses* cubierto contra el número factible de *all-uses*. Una razón de 1 implica que T es adecuado con respecto al criterio *all-uses*.

2. DESCRIPCION DEL PROBLEMA

La Ingeniería en Software, de acuerdo a la norma 610.12 de IEEE, debe aplicar conocimiento científico para el desarrollo, operación y mantenimiento de sistemas software. Para ello cuenta con métodos, técnicas y herramientas para ser utilizadas en cada actividad de acuerdo a las condiciones que se disponga.

Cuando se tienen que tomar decisiones sobre una población a partir de la información tomada de muestras, se dice que se realizan *decisiones estadísticas*. Para realizarlo, lo más útil es tratar de construir hipótesis (o conjeturas) sobre la población a partir de los objetivos de los experimentos. Estas hipótesis pueden ser verdaderas o falsas y se denominan *hipótesis estadísticas*.

Las hipótesis son formuladas con el propósito de ser rechazadas o refutadas. Entonces, si por ejemplo, se quiere decidir cual de dos alternativas es mejor que la otra, se formula la hipótesis "no hay diferencia entre las dos alternativas". Esto querría decir que las diferencias observadas son fluctuaciones del muestreo en la misma población. Este tipo de hipótesis se denomina normalmente *hipótesis nulas* (*null hypotheses*) y se denota como H_0. Toda hipótesis que difiera de una dada se denomina *hipótesis alternativa* y se denota H_i, con i mayor a cero.

La mayoría de los experimentos realizados en Ingeniería del Software formulan suposiciones sobre el tipo de distribución que siguen las variables observadas [5]; [6]; [7]. Pero, para poder realizar esto de forma objetiva deberían contar con una cantidad una gran cantidad de sujetos experimentales (para algunos autores superior a treinta y para otros mas exigentes superior a cien). En contraposición a las suposiciones de distribución, los métodos de análisis no paramétricos no requieren conocer el tipo de distribución al que responden las variables estudiadas [8]; [9]; [10].
En este contexto, a continuación se presenta una revisión estadística mediante un test no paramétricos sobre los resultados experimentales que concluyen, en sus publicaciones originales, que las técnicas de mutación son más efectivas a la hora de detectar errores que el criterio *all-uses*, perteneciente a las técnicas de flujo de datos [11].

3. REVISION DE LOS EXPERIMENTOS

Para determinar si un criterio es mejor que otro, se utilizan las siguientes unidades de medida:

- *Efectividad*, se refiere a la capacidad de detección de fallas de un determinado criterio.
- *Costo*, se refiere al trabajo necesario para satisfacer el criterio.
- *Dificultad de satisfacción*, se refiere a la cantidad de aciertos de un criterio usando conjuntos de prueba de otro criterio.

3.1. Relevamiento de Resultados Experimentales

Se describen en la tabla 2 los estudios empíricos relevados:

DOMINIO	APLICADO A	TÉCNICAS UTILIZADAS	MEDICIONES DOCUMENTADAS	RESULTADOS OBTENIDOS
Comparación de la efectividad en la detección de	Cinco programas distintos:	1- Mutación fuerte 2- Mutación abs/ror	1- Efectividad en la detección de fallas $\frac{N}{} . 100\ \%$	Mutación: Fuerte, abs/ror, 10% — Datos: All-

DOMINIO	APLICADO A	TÉCNICAS UTILIZADAS	MEDICIONES DOCUMENTADAS	RESULTADOS OBTENIDOS			
fallas entre las técnicas de prueba de mutación y de flujo de datos [Mathur A. y Wong W., 1993]	FIND (1)* SORT (2)* STRMATCH (3)* POSITION (2)* STAT (2)* * Cantidad de veces que se ejecutó el programa.	3- Mutación 10% 4- All-uses	*T* *N* = Número de conjunto de pruebas que exponen al menos una falla. *T* = Número total de conjuntos de prueba generados.				uses
				100.00	100.00	86.21	56.67
				100.00	100.00	96.67	96.67
				100.00	76.67	60.00	100.00
				100.00	86.67	96.67	90.00
				93.33	60.00	66.67	60.00
				70.00	53.33	60.00	40.00
				100.00	100.00	93.33	100.00
				100.00	100.00	100.00	100.00
				100.00	100.00	76.67	96.67
				100.00	100.00	73.33	90.00

Tabla 2. Estudios empíricos relevados

3.3. Aplicación de Métodos No Paramétrico

3.3.1. Descripción del Test de Mann-Whitney U o U-Test

El U-Test se aplica cuando hay solamente dos alternativas para la variable independiente Para efectuarlo, se ordenan las observaciones y_{ij} en orden ascendente y se reemplazan por sus rangos R_{ij}, donde el valor de rango 1 se le asigna a la observación más pequeña. En el caso en que haya un empate, se promedia el valor de los rangos para las observaciones empatadas.
Sean R_1 y R_2 la suma de los rangos para cada alternativa; y sean N_1 y N_2, las replicaciones de cada alternativa; entonces la estadística del test estará dada por la siguiente fórmula:

$$U = N_1 N_2 + \frac{N_1(N_1+1)}{2} - R_1 \quad (1)$$

donde la distribución de U es simétrica, y su media y varianza están dadas por:

$$\mu_U = \frac{N_1 N_2}{2} \quad (2)$$

$$\sigma_U^2 = \frac{N_1 N_2 (N_1 + N_2 + 1)}{12} \quad (3)$$

Si N_1 y N_2 son ambas mayores que 7, entonces la distribución de U es aproximadamente normal, de forma que:

$$z = \frac{U - \mu_U}{\sigma_U} \quad (4)$$

está normalmente distribuida con media 0 y varianza 1.

3.3.2. Descripción del Test de Kruskal-Wallis o H-Test

En el caso en que se tengan más de dos alternativas se debe aplicar el H-Test. Primero se deben ordenar las muestras y asignarles un determinado rango al igual que se hace en el U-Test. Sea R_i la suma de los rangos de la observación de la i-ésima técnica, la estadística del H-Test es:

(5) $$H = \frac{1}{S^2}\left(\sum_{i=1}^{a}\frac{R_i^2}{n_i} - \frac{N(N+1)^2}{4}\right)$$

donde *a* es la cantidad de técnicas a comparar, *N* es la cantidad total de mediciones, n_i es la cantidad de mediciones por cada método y la varianza de los rangos S^2 está dada por la siguiente fórmula:

(6) $$S^2 = \frac{1}{N-1}\left(\sum_{i=1}^{a}\sum_{j=1}^{n_i} R_{ij}^2 - \frac{N(N+1)^2}{4}\right)$$

donde R_{ij} es el rango perteneciente al método *i* y a la medicion *j*.

Cuando, para todo *i*, n_i es mayor que 5, *H* tiene una distribución que se aproxima a $\chi^2_{\alpha,a-1}$, si la hipótesis es cierta. Entonces si $H > \chi^2_{\alpha,a-1}$, la hipótesis debe ser rechazada.

3.3.3. Comparación de Todas las Técnicas

Para comparar las cuatro técnicas, se usará el H-Test, cuyo primer paso es la asignación de rangos a las mediciones tomadas:

mutación		**mutación abs/rar**		**mutación 10%**		**All-uses**	
Efectividad (%)	Rango	Efectividad (%)	Rango	Efectividad (%)	Rango	Efectividad (%)	Rango
100.00	31.5	100.00	31.5	86.21	13	56.67	3
100.00	31.5	100.00	31.5	96.67	20.5	96.67	20.5
100.00	31.5	76.67	11.5	60.00	5.5	100.00	31.5
100.00	31.5	86.67	14	96.67	20.5	90.00	15.5
93.33	17.5	60.00	5.5	66.67	8	60.00	5.5
70.00	9	53.33	2	60.00	5.5	40.00	1
100.00	31.5	100.00	31.5	93.33	17.5	100.00	31.5
100.00	31.5	100.00	31.5	100.00	31.5	100.00	31.5
100.00	31.5	100.00	31.5	76.67	11.5	96.67	20.5
100.00	31.5	100.00	31.5	73.33	10	90.00	15.5

Tabla 3. Asignación de rangos a las observaciones

Aplicando las fórmulas (5) y (6), se llega a que $H = 8.32$.

Los valores de χ para distintos valores de significancia α son:

$\chi^2_{0.05,3} = 7.81$	$\chi^2_{0.025,3} = 9.35$	$\chi^2_{0.01,3} = 11.3$

Si bien se podría rechazar la hipótesis para un valor de significancia de 0.05% y afirmar que las técnicas de mutación son mejores que las técnicas de flujo de datos, como para este tipo de test el valor recomendado de significancia es del 0.01% se debe aceptar la hipótesis. Por lo tanto, podemos afirmar que *no existe evidencia para afirmar que las técnicas de mutación son mejores que las de análisis de flujo de datos*.

A continuación se va a efectuar una comparación de las técnicas de a pares, para intentar obtener diferencia puntuales.

3.3.4. Comparación de las Técnicas agrupadas de a dos

A continuación se comparan las técnicas mediante el U-Test. Si se reemplazan las variables por sus respectivos valores, la fórmula (4) se transforma en:

$$z = \frac{105 - R_1}{13.23}$$

La hipótesis de que no hay diferencia en los resultados con ambas técnicas se rechazará si el módulo de *z* es mayor que 1.96, lo que equivale a decir que el nivel de significancia es del 5%.

1) Mutación fuerte vs. mutación abs/ror .

mutación		mutación abs/ror	
Efectividad (%)	Rango	Efectividad (%)	Rango
100.00	13.50	100.00	13.50
100.00	13.50	100.00	13.50
100.00	13.50	76.67	4.00
100.00	13.50	86.67	5.00
93.33	6.00	60.00	2.00
70.00	3.00	53.33	1.00
100.00	13.50	100.00	13.50
100.00	13.50	100.00	13.50
100.00	13.50	100.00	13.50
100.00	13.50	100.00	13.50

Tabla 4. Asignación de rangos a las observaciones

$R_1 = 117$
$R_2 = 93$

Por lo tanto; $z = -0.907$ y no se puede rechazar la hipótesis. Es decir, que en este caso no existe evidencia de que la mutación fuerte es más efectiva que la mutación abs/ror.

2) Mutación fuerte vs. mutación 10%.

Mutación		mutación 10%	
Efectividad (%)	Rango	Efectividad (%)	Rango
100.00	16.00	86.21	7.00
100.00	16.00	96.67	10.50
100.00	16.00	60.00	1.50
100.00	16.00	96.67	10.50
93.33	8.50	66.67	3.00
70.00	4.00	60.00	1.50
100.00	16.00	93.33	8.50
100.00	16.00	100.00	16.00

Mutación		mutación 10%	
100.00	16.00	76.67	6.00
100.00	16.00	73.33	5.00

Tabla 5. Asignación de rangos a las observaciones

$R_1 = 140.5$
$R_2 = 69.5$
Por lo tanto; $z = -2.68$ y se puede rechazar la hipótesis. Es decir, que en este caso existe evidencia de que la mutación fuerte es más efectiva que la mutación 10%.

3) Mutación fuerte vs. all-uses.

Mutación		All-uses	
Efectividad (%)	Rango	Efectividad (%)	Rango
100.00	15.00	56.67	2.00
100.00	15.00	96.67	8.50
100.00	15.00	100.00	15.00
100.00	15.00	90.00	5.50
93.33	7.00	60.00	3.00
70.00	4.00	40.00	1.00
100.00	15.00	100.00	15.00
100.00	15.00	100.00	15.00
100.00	15.00	96.67	8.50
100.00	15.00	90.00	5.50

Tabla 6. Asignación de rangos a las observaciones

$R_1 = 131$
$R_2 = 79$

Por lo tanto; $z = -1.965$. En este caso el valor de z está apenas (0.005) por encima del umbral del 5% de significancia. La hipótesis, de todas formas, puede ser rechazada y puede afirmarse que la mutación fuerte es más efectiva que el criterio de all-uses.

4) Mutación abs/ror vs. mutación 10%.

mutación abs/ror		Mutación 10%	
Efectividad (%)	Rango	Efectividad (%)	Rango
100.00	17.00	86.21	9.00
100.00	17.00	96.67	12.50
76.67	7.50	60.00	3.00
86.67	10.00	96.67	12.50
60.00	3.00	66.67	5.00
53.33	1.00	60.00	3.00
100.00	17.00	93.33	11.00
100.00	17.00	100.00	17.00
100.00	17.00	76.67	7.50
100.00	17.00	73.33	6.00

Tabla 7. Asignación de rangos a las observaciones

$R_1 = 123.5$
$R_2 = 83.5$

Por lo tanto; $z = -1.39$ y no se puede rechazar la hipótesis. Es decir, que en este caso no existe evidencia de que la mutación abs/ror es más efectiva que la mutación 10%.

5) Mutación abs/ror vs. all-uses.

mutación abs/ror		All-uses	
Efectividad (%)	Rango	Efectividad (%)	Rango
100.00	16.00	56.67	3.00
100.00	16.00	96.67	10.50
76.67	6.00	100.00	16.00
86.67	7.00	90.00	8.50
60.00	4.50	60.00	4.50
53.33	2.00	40.00	1.00
100.00	16.00	100.00	16.00
100.00	16.00	100.00	16.00
100.00	16.00	96.67	10.50
100.00	16.00	90.00	8.50

Tabla 8. Asignación de rangos a las observaciones

$R_1 = 115.5$
$R_2 = 94.5$
Por lo tanto; $z = -0.79$ y no se puede rechazar la hipótesis. Es decir, que en este caso no existe evidencia de que la mutación abs/ror es más efectiva que el criterio de all-uses.

6) Mutación 10% vs. all-uses.

mutación 10%		All-uses	
Efectividad (%)	Rango	Efectividad (%)	Rango
86.21	9.00	56.67	2.00
96.67	14.50	96.67	14.50
60.00	4.00	100.00	18.50
96.67	14.50	90.00	10.50
66.67	6.00	60.00	4.00
60.00	4.00	40.00	1.00
93.33	12.00	100.00	18.50
100.00	18.50	100.00	18.50
76.67	8.00	96.67	14.50
73.33	7.00	90.00	10.50

Tabla 9. Asignación de rangos a las observaciones

$R_1 = 97.5$
$R_2 = 112.5$

Por lo tanto; $z = 0.567$ y no se puede rechazar la hipótesis. Es decir, que en este caso no existe evidencia de que la mutación 10% es más efectiva que el criterio de all-uses.

4. CONCLUSIONES

La experimentación de software trata de buscar reglas empíricas que proporcionen evidencias sobre las ventajas o desventajas de los distintos métodos, técnicas o herramientas empleadas en la construcción de sistemas software. Si no se realiza una revisión formal sobre los resultados de los experimentos se corre el riesgo de sostener conclusiones erróneas. Además, es posible también obtener resultados incorrectos si se estiman parámetros sobre distribuciones estadísticas que fueron asumidas erróneamente.

En este contexto, la aplicación de los test no paramétricos a demostrado que si bien a simple vista los valores obtenidos parecían indicar que las técnicas de mutación eran mejores que las de flujo de datos, no existe evidencia empírica que justifique esta afirmación. Las únicas diferencias encontradas como resultado de la aplicación del test son las siguientes:

- La técnica de mutación fuerte es mas efectiva que la técnica de mutación 10%
- La técnica de mutación fuerte es mas efectiva que la técnica all-uses

Por lo expuesto, no puede afirmarse que las técnicas de mutación, en forma general, son mejores que las técnicas de flujo de datos.

5. REFERENCIAS

[1] Juristo N., Moreno A. y Vegas S. 2003. *Limitations of Empirical Testing Technique Knowledge*. Lecture notes on empirical software engineering archive, pages 1-38. World Scientific Publishing Co., Inc. River Edge, NJ, USA.

[2] Beizer, B. 1990. Software Testing Techniques. International Thomson Computer Press.

[3] Myers, G. 1979. The Art of Software Testing. Wiley-interscience.

[4] MONTHRA, 1987. *The Mothra Software Testing Environment User's Manual*. Technical Report SERC-TR-4-P, Software Engineering Research Center, Purdue University.

[5] Bieman, J. yb Schultz, J., 1992. *An Empirical Evaluation (and specification) of the All du-paths Testing Criterion*. Software Engineering Journal. Pages 43-51, January.

[6] Frankl, P. y Iakounenko, O., 1998. *Further Empirical Studies of Test Effectiveness*. In Proceedings of the ACM SIGSOFT International Symposium on Foundations on Software Engineering, pages 153-162, Lake Buena Vista, Florida, USA.

[7] Juristo N. y Moreno A. 2001. *Basics of Software Engineering Experimentation*. Kluwer Academic Publisher. Dordrecht.

[8] Ledesma, D. 1980. *Estadística Médica*. Eudeba. Buenos Aires.

[9] Montgomery, D. y Runger, G. 2002. *Probabilidad y Estadística*. Limusa Wiley. Mexico DF.

[10] García, R. 2004. *Inferencia Estadísitica y Diseño de Experimentos*. Eudeba. Buenos Aires.

[11] Mathur A. y Wong W., 1993. *Comparing the fault detection effectiveness of mutation and data flow testing: An empirical study*. Tech. Report SERC-TR-146-P, Software Engineering Research Center.