Evaluación de la calidad de los mutantes en la prueba de mutaciones

Antonia Estero Botaro, Juan José Domínguez Jiménez, Lorena Gutiérrez Madroñal e Inmaculada Medina Bulo

Dpto. de Lenguajes y Sistemas Informáticos,
Grupo de investigación UCASE, Universidad de Cádiz
{antonia.estero,juanjose.dominguez,lorena.gutierrez,inmaculada.medina}@uca.es

Resumen La prueba de mutaciones es una técnica de prueba basada en fallos computacionalmente costosa. Debido a esto, se han propuesto diferentes técnicas para reducir dicho coste computacional mediante la reducción del número de mutantes generados. La efectividad de estas técnicas se podría medir de acuerdo a la calidad del subconjunto de mutantes generados. Para ello es necesario establecer una métrica que determine la calidad de un mutante frente al conjunto de casos de prueba sobre el que se ejecuta. En este artículo se presenta una métrica de la calidad de los mutantes, así como una métrica de la calidad de los operadores de mutación. Además se muestran los resultados experimentales obtenidos al aplicar dichas métricas a varias composiciones WS-BPEL.

Keywords: prueba de mutaciones, composición de servicios, servicios web, WS-BPEL, evaluación de la calidad

1. Introducción

La prueba de mutaciones [14] es una técnica de prueba que introduce fallos simples en el programa a probar mediante la aplicación de los *operadores de mutación*. Como resultado obtenemos nuevos programas denominados *mutantes*. El programa original y los mutantes se ejecutan frente al conjunto de casos de prueba, si la salida de un mutante para un caso de prueba difiere de la del programa original, se dice que el mutante ha *muerto* lo que significa que dicho caso de prueba ha sido capaz de detectar el fallo introducido en el mutante. Si la salida de ambos es la misma para todos los casos de prueba decimos que el mutante permanece *vivo*.

Cada operador de mutación se corresponde con una categoría de error típico que el desarrollador podría cometer. Así, si un programa contiene la instrucción $x > 1$ y disponemos de un operador de mutación que actúa sobre los operadores relacionales (cambia un operador relacional por otro), se podrían producir los mutantes que contengan: $x < 1$, $x = 1$, etc.

Uno de los principales inconvenientes de la prueba de mutaciones es el coste computacional que supone la ejecución de la gran cantidad de mutantes que

se generan a partir del programa original. Esto se debe a que, normalmente, disponemos de un elevado número de operadores de mutación, que generan un gran número de mutantes que deben ser ejecutados contra el conjunto de casos de prueba [14]. Para reducir el tiempo de ejecución se han propuesto diversos tipos de técnicas que son recopiladas por Jia y Harman [10] y por Polo y Reales [16]. Algunas de estas técnicas se basan en la reducción del número de mutantes que se genera, así tenemos: *muestreo de mutantes* [1,4], *agrupamiento de mutantes* [9], *mutación selectiva* [15], *mutación de orden superior* [10] y *mutación evolutiva* [6].

Otro inconveniente de la prueba de mutaciones es la existencia de *mutantes equivalentes*. Estos tienen el mismo comportamiento que el programa original, es decir, siempre producen la misma salida que este. La identificación de los mutantes equivalentes se suele realizar manualmente, siendo una tarea costosa.

El objetivo de este trabajo es definir una métrica de la calidad de los mutantes y de los operadores de mutación. Para poder evaluar la eficacia de la métrica planteada, esta será aplicada en diversas composiciones de servicios WS-BPEL. ¿Por qué WS-BPEL? Como consecuencia de la evolución del software hacia arquitecturas orientadas a servicios, se define un lenguaje que facilita la composición de servicios web (SW), el estándar OASIS WS-BPEL 2.0 [13], que es una referencia a nivel industrial. WS-BPEL permite crear nuevos SW que modelan procesos de negocio más complejos mediante el uso de SW preexistentes. Surge la necesidad de profundizar en el estudio de las pruebas de este tipo de software debido al rápido crecimiento del impacto económico de las composiciones de servicios WS-BPEL.

La métrica de calidad de los mutantes permitirá estudiar la efectividad de las técnicas de reducción de mutantes en función de la calidad del subconjunto de mutantes que generan, y por tanto, establecer su grado de eficacia basándonos en la calidad de los resultados que obtienen. Por otro lado, la métrica de calidad de los operadores de mutación permitirá determinar cuales son los operadores más interesantes para un lenguaje.

El resto del artículo presenta la siguiente estructura: la sección 2 introduce las principales características del lenguaje WS-BPEL. La sección 3 presenta los operadores de mutación definidos para WS-BPEL. La sección 4 define formalmente la calidad de los mutantes y de los operadores de mutación. La sección 5 presenta el método experimental y las herramientas empleadas. La sección 6 presenta los resultados obtenidos para cuatro composiciones WS-BPEL en relación a la calidad de los mutantes y de los operadores de mutación. La sección 7 muestra los principales trabajos que tratan el tema de la calidad en la prueba de mutaciones. Finalmente, la sección 8 presenta las conclusiones y líneas futuras de trabajo.

2. Lenguaje WS-BPEL

WS-BPEL es un lenguaje basado en XML que permite especificar el comportamiento de un proceso de negocio basado en interaciones con SW [13]. Los principales elementos constructivos de un proceso WS-BPEL son las actividades,

estas pueden ser de dos tipos: *básicas*, que son las que realizan una determinada labor (recepción de un mensaje, manipulación de datos, etc.) y *estructuradas*, que pueden contener otras actividades y definen la lógica de negocio.

A las actividades pueden asociarse un conjunto de atributos y de contenedores. Los contenedores pueden incluir diferentes elementos, que a su vez pueden tener atributos asociados. Veamos un ejemplo:

```
<flow>  ← Actividad estructurada
  <links>  ← Contenedor
    <link name="comprobVuelo-A-reservVuelo"  ← Atributo/>  ← Elemento
  </links>
  <invoke name="comprobVuelo" ... >  ← Actividad básica
    <sources>  ← Contenedor
      <source linkName="comprobVuelo-A-reservVuelo" ← Atributo/> ← Elemento
    </sources>
  </invoke>
  <invoke name="comprobHotel" ... />
  <invoke name="comprobAlquilerCoche" ... />
  <invoke name="reservVuelo" ... >
    <targets>   ← Contenedor
      <target linkName="comprobVuelo-A-reservVuelo" />  ← Elemento
    </targets>
  </invoke>
</flow>
```

WS-BPEL proporciona actividades para la recepción de mensajes (`receive` y `pick`), para el envío de mensajes (`reply`), para definir la estructura lógica del proceso (`sequence`, `flow`, `if`, ...). Además, WS-BPEL permite realizar acciones en paralelo y de forma sincronizada. Un ejemplo de ello es la actividad `flow`, mediante la cual se puede especificar un conjunto de actividades que se van a realizar concurrentemente, así como las condiciones de sincronización entre ellas.

3. Operadores de Mutación para WS-BPEL

Un aspecto clave para la adecuada aplicación de la prueba de mutaciones es la elección de los operadores que mutación, que deben ser diseñados de forma específica para cada lenguaje de programación [2]. Cada operador de mutación representa un error típico que puede cometer el programador, o bien, puede forzar un determinado criterio de cobertura.

Esta sección describe los operadores de mutación definidos por Estero y col. [7] para el lenguaje WS-BPEL 2.0. La tabla 1 muestra el nombre y una descripción breve de cada uno de los operadores, clasificados por categorías. Se han marcado con ☆ los operadores específicos de WS-BPEL 2.0 que no aparecen en otros lenguajes. Boubeta y col. [3] han realizado un estudio comparativo entre los operadores definidos para WS-BPEL 2.0 y los definidos para otros lenguajes, concluyendo que el 50 % de los operadores definidos para WS-BPEL son específicos de este lenguaje.

Operador	Descripción
	Mutación de identificadores
ISV	Sustituye el identificador de una variable por el de otra del mismo tipo
	Mutación de expresiones
EAA	Sustituye un operador aritmético (`+`, `-`, `*`, `div`, `mod`) por otro del mismo tipo
EEU	Elimina el operador `-` unario de cualquier expresión
ERR	Sustituye un operador relacional (`=`, `!=`, `<`, `>`, `<=`, `>=`) por otro del mismo tipo
ELL	Sustituye un operador lógico (`and`, `or`) por otro del mismo tipo
ECC	Sustituye un operador de camino (`/`, `//`) por otro del mismo tipo
ECN	Modifica una constante numérica incrementándola/decrementándola en una unidad, o añadiendo/eliminando un dígito
EMD	Modifica una expresión de duración, reemplazándola por 0 o por la mitad de su valor inicial
EMF	☆ Modifica una expresión de fecha límite, reemplazándola por la fecha actual.
	Mutación de actividades concurrentes
ACI	☆ Cambia el atributo `createInstance` de una actividad de recepción de mensajes a *no*
AFP	☆ Sustituye una actividad `forEach` secuencial por una paralela
ASF	☆ Sustituye una actividad `sequence` por una actividad `flow`
AIS	☆ Cambia el atributo `isolated` de un `scope` a *no*
	Mutación de actividades no concurrentes
AEL	Borra una actividad
AIE	Borra un elemento `elseif` o el elemento `else` de una actividad `if`
AWR	Sustituye una actividad `while` por una actividad `repeatUntil` y viceversa
AJC	☆ Borra el atributo `joinCondition` de una actividad
ASI	☆ Intercambia el orden de dos actividades hijas de una actividad `sequence`
APM	☆ Borra un elemento `onMessage` de una actividad `pick`
APA	☆ Borra el elemento `onAlarm` de una actividad `pick` o de un manejador de eventos
	Mutación de condiciones excepcionales y eventos
XMF	Borra un elemento `catch` o el elemento `catchAll` de un manejador de fallos
XMC	☆ Borra la definición de un manejador de compensación
XMT	☆ Borra la definición de un manejador de terminación
XTF	Sustituye el fallo lanzado por una actividad `throw`
XER	☆ Borra una actividad `rethrow`
XEE	☆ Borra un elemento `onEvent` de un manejador de eventos

Tabla 1. Operadores de mutación para WS-BPEL 2.0

4. Definición Formal de la Calidad de un Mutante

A continuación se presenta una formalización de una porción significativa de la teoría de la prueba de mutaciones. El objetivo es doble, por un lado se desarrolla una notación consistente y clara para representar varios conceptos cuyas definiciones están dispersas, faltan o incluso aparecen de formas ligeramente diferentes en la bibliografía. Por otro lado, se introducen las definiciones de mutante resistente, mutante débil, mutante difícil de matar, calidad de un mutante y calidad de un operador de mutación. Esta formalización permite razonar de forma precisa sobre todos estos conceptos.

4.1. Nociones Básicas

Sea p un programa a probar. Un mutante de primer orden de p es una variante simple obtenida modificando, insertando o borrando una unidad sintáctica simple de p. Se definen los siguientes conjuntos:

- I_p: El espacio de entrada de p.
- T_p: Un conjunto de casos de prueba (*test-suite*) para p.
- M_p: Un conjunto de mutantes válidos [1] de p.

Sea $m \in M_p$ y $t \in T_p \subseteq I_p$.[2]. Se define también la siguiente relación, donde $q(t)\downarrow$ significa que el programa q se para en el caso de prueba t, mientras que $q(t)\uparrow$ tiene el significado contrario.

Definición 1. *Mutante equivalente bajo un caso de prueba t.*

$$m(t) \approx p(t) \equiv (m(t)\downarrow \wedge p(t)\downarrow \wedge m(t) = p(t)) \vee (m(t)\uparrow \wedge p(t)\uparrow) \quad (1)$$

Esto significa que un mutante m es equivalente a su programa original p bajo el caso de prueba t si, una vez terminada su ejecución, ambos producen la misma salida o si ninguno de ellos termina. Las condiciones de terminación pueden ser relajadas de diversas formas, proporcionando nociones diferentes de equivalencia. Sin embargo, la formalización desarrollada en este trabajo es altamente independiente de este aspecto.

En el resto de la sección, se omitirá la dependencia de p, considerando todos los conceptos definidos implícitamente para un determinado programa a probar. Además, se definirán varias nociones dependientes de I, T o M. Para una mejor legibilidad, también se omitirán estas dependencias de la notación, ya que estarán claras a partir de su contexto.

[1] Los operadores de mutación pueden producir mutantes no válidos, es decir, mutantes sintácticamente incorrectos, o bien, debido a la estricta semática de WS-BPEL es posible que un mutante válido sintácticamente pueda ser semáticamente no válido. Dado que los mutantes no válidos generados no se ejecutan sólo consideraremos el conjunto de mutantes de p que se puede ejecutar.

[2] Los casos de prueba se definen sólo como entradas, ya que en la prueba de mutaciones el programa original actúa como oráculo.

Definición 2. *Matando un mutante.*

$$t \text{ mata } m \equiv m(t) \not\approx p(t) \tag{2}$$

Definición 3. *Los casos de prueba que matan al mutante m.*

$$K_m = \{t \in T \mid t \text{ mata } m\} \tag{3}$$

Por tanto, obviamente, $\varnothing \subseteq K_m \subseteq T$.

Definición 4. *Mutantes muertos.*

$$D = \{m \in M \mid \exists t \in T \; t \text{ mata } m\} \tag{4}$$

Definición 5. *Mutantes vivos.*

$$V = \{m \in M \mid \neg\exists t \in T \; t \text{ mata } m\} \tag{5}$$

$$D = \{m \in M \mid K_m \neq \varnothing\} \qquad V = \{m \in M \mid K_m = \varnothing\}$$

Por tanto, se tiene que $M = D \cup V$, $D \cap V = \varnothing$, $D = M - V$, y $V = M - D$.

Definición 6. *Mutantes equivalentes.*

$$E = \{m \in M \mid \neg\exists t \in I \; t \text{ mata } m\} \tag{6}$$

Por tanto, obviamente, $E \subseteq V \subseteq M$.

Definición 7. *Puntuación de mutación.*
La calidad de los conjuntos de los casos de prueba se mide mediante la puntuación de mutación, que indica la proporción de mutantes muertos frente al número de mutantes no equivalentes.

$$\mathcal{S} = \frac{|D|}{|M| - |E|} \tag{7}$$

4.2. Adecuación y Redundancia de Conjuntos de Casos de Prueba

Definición 8. *T es* adecuado *para M si todo mutante no equivalente en M es matado por algún caso de prueba en T.*

$$T \textit{ es adecuado para } M \textit{ si } \forall m \in M - E \; \exists t \in T \; t \text{ mata } m \tag{8}$$

Nota. Cuando T es adecuado, $P = E$ y $D = M - E$. Además, siendo $E \subseteq M$, está claro que $|D| = |M| - |E|$. Por tanto, $\mathcal{S} = 1$ para conjuntos de casos de prueba adecuados.

Definición 9. *Mutantes que cubren un caso de prueba t.*

$$C_t = \{m \in M \mid t \text{ mata } m\} \tag{9}$$

Por tanto, obviamente, $\varnothing \subseteq C_t \subseteq M$. Por supuesto, esta definición puede ser extendida a conjuntos de casos de prueba de una forma natural:

$$C_T = \bigcup_{t \in T} C_t \tag{10}$$

Definición 10. *Un caso de prueba t es* redundante *con respecto a un conjunto de casos de prueba T si* $C_T = C_{T \cup \{t\}}$. *T es* no redundante *si no contiene ningún caso de prueba t tal que t es redundante con respecto a* $T - \{t\}$.

4.3. Resistencia de los Mutantes

Tradicionalmente, la prueba de mutaciones clasifica a los mutantes una vez ejecutados frente a los casos de prueba como vivos o muertos. Sin embargo, para poder estudiar la calidad de los mutantes consideramos necesario clasificar a los mutantes muertos en función del número de casos de prueba que los mata. Para ello proponemos los conceptos de *mutante débil*, *mutante resistente* y *mutante difícil de matar*, cuyas definiciones formales se muestran a continuación.

Definición 11. *Un mutante es* débil *cuando lo matan todos los casos de prueba.*

$$W = \{m \in M \mid K_m = T\} \tag{11}$$

Definición 12. *Un mutante es* resistente *cuando lo mata un único caso de prueba.*

$$R = \{m \in M \mid |K_m| = 1\} \tag{12}$$

Definición 13. *Un mutante resistente es* difícil de matar *cuando cubre únicamente el caso de prueba que lo mata.*

$$H = \{m \in R \mid \forall t \in K_m \ \ |C_t| = 1\} \tag{13}$$

4.4. Calidad de un Mutante

Teniendo en cuenta que los mutantes resistentes son más interesantes que los débiles, ya que necesitan casos de prueba más específicos para detectarlos, los mutantes difíciles de matar serían los más interesantes desde este punto de vista. Se propone una métrica de la calidad de un mutante que va a depender del número de casos de prueba que lo matan y del número de mutantes que cubren dichos casos de prueba. Al tratarse de una medida dependiente de los casos de prueba de que disponemos, se ha considerado importante que el conjunto de casos de prueba utilizado cumpla dos propiedades, debe ser adecuado (8) y no redundante (Definición 10).

Definición 14. *Calidad de un mutante*

$$Qm_i = 1 - \frac{\sum_{t \in K_m} |C_t|}{(|M| - |E|) \cdot |T|}, \qquad \forall m_j \in D \tag{14}$$

$$Qm_j = 0, \qquad \forall m_j \in E \tag{15}$$

Teniendo en cuenta la definición de calidad de un mutante propuesta, podemos analizar el posible comportamiento de los mutantes respecto a ella.

Los mutantes menos significativos de cara a una selección de mutantes para determinar el correcto funcionamiento de un programa son los mutantes débiles, es decir, aquéllos que son matados por todos los casos de prueba. Si disponemos de un conjunto de mutantes formado únicamente por mutantes débiles, la calidad de cada uno de estos mutantes será 0.

Los mutantes más interesantes van a ser los resistentes, y más en concreto los mutantes difíciles de matar. Si m_h es un mutante difícil de matar, su calidad, según (13), será $Qm_h = 1 - 1/((|M| - |E|)\,|T|)$.

Un mutante resistente es aquel que es matado por un solo caso de prueba. Podemos establecer varios grados de resistencia teniendo en cuenta el número de mutantes adicionales que mata dicho caso de prueba. Así, el peor mutante resistente será aquel para el que el caso de prueba, t, que lo mata a él mata también al resto de mutantes, es decir, $|Ct| = |D| = |M| - |E|$. Por tanto, si m_r es el peor resistente, su calidad es $Qm_r = 1 - (|M| - |E|)/((|M| - |E|)\,|T|) = 1 - 1/|T|$. De igual forma podríamos establecer otros valores umbrales para la calidad de un mutante resistente, por ejemplo que mate a la mitad de los mutantes, que mate a la cuarta parte, etc.

4.5. Calidad de un Conjunto de Mutantes

La definición de la calidad de un mutante se puede extender a cualquier conjunto de mutantes.

Definición 15. *Si M es el conjunto de mutantes de un programa p, y E es el conjunto de mutantes equivalentes que se generan.*

$$QM = \frac{Qm_1 + Qm_2 + \ldots + Qm_n}{|M|} \ \forall m_i \in M \tag{16}$$

que según la definición de calidad de un mutante (14) resulta ser:

$$QM = 1 - \frac{\sum_{m \in M} \sum_{t \in K_m} |C_t|}{(|M| - |E|)\,|M|\,|T|} \tag{17}$$

Esta definición permite establecer la calidad de los mutantes que son generados por un operador de mutación. Así, esta medida nos puede servir para evaluar la calidad de un operador de mutación.

Definición 16. *Calidad de un operador de mutación*
Si M_O es el conjunto de mutantes que genera el operador O.

$$QM_o = \frac{Qm_1 + Qm_2 + \ldots + Qm_n}{|M_O|} \ \forall m_i \in M_O \tag{18}$$

que según la definición de calidad de un mutante (14) resulta ser:

$$QM_o = 1 - \frac{\sum\limits_{m \in M_O} \sum\limits_{t \in K_m} |C_t|}{(|M_O| - |E_O|)\,|M_O|\,|T|} \tag{19}$$

5. Método Experimental

Hemos usado cuatro composiciones WS-BPEL: Préstamo (P), CompraVenta (CV), SumaCuadrados (SC) y AgenciaViajes (AV). Las dos primeras se han tomado del estándar WS-BPEL 2.0 [13] y la cuarta de Netbeans [12], aunque han sido modificadas para que se puedan aplicar más operadores de mutación.

Para calcular la calidad de los mutantes generados por cada una de las composiciones hemos utilizado el procedimiento que se detalla a continuación:

1. Generar todos los mutantes posibles.
2. Ejecutar los mutantes frente a un conjunto de casos de prueba inicial. Esta ejecución nos permite conocer qué mutantes son no válidos y determinar, inspeccionándolos, cuáles de los mutantes que sobreviven son no equivalentes.
3. Completar, si es necesario, el conjunto de casos de prueba inicial con casos adicionales que maten a los mutantes no equivalentes. Este paso y el previo se repiten hasta conseguir un conjunto de casos de prueba adecuado.
4. Reducir el conjunto de casos de prueba adecuado para que sea no redundante.
5. Calcular la calidad de cada mutante, el porcentaje de mutantes de calidad que genera una composición y la calidad de los operadores en función de la calidad de los mutantes que producen.

5.1. Herramientas Desarrolladas

MuBPEL [11] es una herramienta de generación de mutantes, que además los ejecuta frente a un conjunto de casos de prueba y compara su salida con la de la composición original. Consta de tres componentes principales: un analizador, un generador de mutantes y el motor de ejecución. MutMix es una aplicación que nos facilita el cálculo de las calidades de los mutantes y de los operadores que los generan, también nos indica los posibles casos de prueba redundantes que se hayan utilizado en la composición estudiada, y si existen mutantes difíciles de matar o un conjunto de estos.

6. Resultados

Esta sección muestra los resultados obtenidos en los estudios realizados sobre calidad de los mutantes y calidad de los operadores de mutación.

La tabla 2 muestra las características de las cuatro composiciones utilizadas y el tamaño del conjunto de casos de prueba final ($|T|$).

	P	CV	SC	AV
Líneas de código	110	202	96	384
Mutantes totales	62	41	48	217
Mutantes equivalentes	7	3	4	59
Mutantes no válidos	2	2	2	3
$\|T\|$	4	4	1	11

Tabla 2. Composiciones utilizadas

La tabla 3 muestra los porcentajes de mutantes de calidad obtenidos para cada composición con distintos valores umbrales de calidad. Como se ha indicado en la sección 4 podemos establecer que un mutante es de calidad si es resistente y si el único caso de prueba que lo mata, también mata a un cierto porcentaje de mutantes. Hemos calculado los porcentajes de mutantes de calidad que se obtienen para valores umbrales de calidad de $1 - 1/T$, $1 - 3/4T$, $1 - 1/2T$ y $1 - 1/4T$, es decir, cuando un mutante es resistente el caso de prueba que lo cubre también mata al 100 %, al 75 %, al 50 % y al 25 % de los mutantes muertos.

Los resultados obtenidos muestran, como es de esperar, que a medida que el umbral es más restrictivo el porcentaje de mutantes de calidad disminuye. La composición SC, para la que se obtiene un $|T| = 1$, no produce ningún mutante de calidad. Esto es debido a que la composición se demasiado simple y existe un caso de prueba que mata a todos los mutantes. Las otras dos composiciones que producen un número de mutantes pequeño, P y CV, se comportan de una forma similar; obteniendo un número alto de mutantes de calidad para los umbrales $1-1/T$ y $1-3/4T$, un porcentaje entre el 10 % y el 15 % para el umbral $1-1/2T$, y un 0 % de mutantes de calidad para $1-1/4T$. La composición que genera más mutantes, AV, presenta un porcentaje alto de mutantes de calidad para los tres primeros umbrales y para el último obtiene un valor cercano al 10 %.

El estudio del porcentaje de mutantes de calidad que genera una composición nos puede servir para comparar la efectividad de distintas técnicas de reducción de mutantes. Así, aquellas técnicas que produzcan un mayor porcentaje de mutantes de calidad dentro del subconjunto de mutantes generados tendrán una mayor efectividad que las que producen menos. Según los resultados obtenidos, si las composiciones sobre las que se trabaja producen un número bajo de mutantes, el valor umbral de calidad que se debe exigir será de $1 - 1/2T$. Si las composiciones generan un mayor número de mutantes el umbral de calidad se puede establecer en $1 - 1/4T$. Teniendo en cuenta que las técnicas de reducción

	Mutantes de calidad (%)			
	$1-1/T$	$1-3/4T$	$1-1/2T$	$1-1/4T$
P	53,2	33,9	11,3	0
CV	63,4	58,5	9,8	0
SC	0	0	0	0
AV	51,2	47,0	44,7	11,1

Tabla 3. Clasificación de los mutantes

de mutantes generan entre el 25 % y el 10 % del número total de mutantes, los porcentajes de mutantes de calidad a obtener son razonables.

La tabla 4 muestra la calidad de los operadores de mutación utilizados en las composiciones P, CV y AV. Para este estudio no hemos utilizado la composición SC dado que no presenta ningún mutante de calidad. La columna Q_{O-E} muestra el valor de la calidad del operador sin contabilizar los mutantes equivalentes que genera, mientras que Q_O sí los contabiliza.

Tomando como referencia el mínimo valor exigido a la calidad de un mutante en el estudio realizado anteriormente para el umbral $1-1/2T$, establecemos como valor mínimo de calidad de los operadores $Q_{O-E} = 0,87$. Los mutantes que obtienen este valor de calidad son mutantes resistentes para los que el caso de prueba que los cubre, mata al 50 % del resto de mutantes.

Los operadores que obtienen valores de $Q_{O-E} \geq 0,87$ son AIE, APA, XMT, XTF, XER y XEE, y además no producen mutantes equivalentes puesto que $Q_O = Q_{O-E}$, excepto APA que produce un pequeño porcentaje. Por tanto, los consideramos de alta calidad. Sin embargo, XMT, XTF, XER y XEE sólo son aplicados en una de las composiciones luego requerirían un estudio más amplio.

Los operadores ERR, ELL, ECN, AEL, AJC, ASI y XMC obtienen una media de Q_{O-E} ligeramente inferior a 0,87, sin embargo los consideramos de alta calidad porque apenas generan mutantes equivalentes ($Q_{O-E} - Q_O < 0,1$).

Los operadores ECC, EMD, EMF y XMF en principio se podrían considerar de alta calidad porque su $Q_{O-E} \geq 0,87$. Sin embargo, $Q_{O-E} - Q_O > 0,2$ porque producen muchos mutantes equivalentes. ECC, que es un operador de mutación de caminos, produce muchos mutantes equivalentes en las composiciones utilizadas porque los caminos que se emplean son simples y no permiten generar, normalmente, mutantes no equivalentes. EMD, EMF y XMF requieren una revisión para intentar reducir los mutantes equivalentes que producen.

Los operadores ASF y APM son los que consideramos de peor calidad. Casi todos los mutantes que genera ASF son equivalentes. Este operador sustituye una actividad `sequence` por `flow` por lo que a menos que exista una dependencia de datos entre las actividades implicadas los mutantes producidos serán equivalentes. Podemos ver que presenta un valor $Q_O = 0$ para dos de las composiciones estudiadas. Por otro lado, el operador APM no genera mutantes equivalentes pero obtiene valores de Q_O y Q_{O-E} bajos. Estos dos operadores requieren un estudio con otras composiciones para definitivamente descartarlos.

	P			CV			AV		
	M_O	Q_O	Q_{O-E}	M_O	Q_O	Q_{O-E}	M_O	Q_O	Q_{O-E}
ERR	10	0,77	0,77	5	0,83	0,83	30	0,70	0,87
ELL	-	-	-	3	0,80	0,80	-	-	-
ECC	13	0,38	0,82	-	-	-	24	0,04	0,96
ECN	4	0,79	0,79	-	-	-	24	0,93	0,93
EMD	-	-	-	2	0,76	0,76	6	0,49	0,98
EMF	-	-	-	-	-	-	2	0,49	0,98
ASF	4	0,79	0,79	2	0,00	-	14	0,00	-
AEL	17	0,77	0,77	16	0,79	0,79	62	0,87	0,94
AIE	2	0,85	0,85	1	0,90	0,90	-	-	-
AJC	-	-	-	1	0,84	0,84	-	-	-
ASI	12	0,73	0,73	2	0,84	0,84	34	0,73	0,92
APM	-	-	-	4	0,68	0,68	-	-	-
APA	-	-	-	1	0,84	0,84	3	0,64	0,96
XMF	-	-	-	2	0,42	0,84	5	0,95	0,95
XMC	-	-	-	2	0,84	0,84	-	-	-
XMT	-	-	-	-	-	-	1	0,89	0,89
XTF	-	-	-	-	-	-	6	0,97	0,97
XER	-	-	-	-	-	-	4	0,97	0,97
XEE	-	-	-	-	-	-	2	0,98	0,98

Tabla 4. Calidad de los operadores de mutación

6.1. Limitaciones del Estudio

No se ha podido evaluar la calidad de todos los operadores de mutación definidos para WS-BPEL, ya que todos ellos no son aplicables a cualquier composición y no existen muchas composiciones WS-BPEL de libre disposición con las que poder hacer pruebas. Las composiciones utilizadas han sido creadas de forma expresa o modificadas para ejercitar un mayor número de operadores de mutación. Sería necesario disponer de más composiciones y más grandes para poder evaluar todos los operadores de mutación definidos.

7. Trabajos Relacionados

Derezińska [5] ha evaluado la calidad de los operadores de mutación definidos para C#. Para ello calcula la efectividad de un conjunto de casos de prueba para un operador como el cociente entre el número de ejecuciones de casos de prueba que matan mutantes generados por ese operador frente al número total de ejecuciones realizadas sobre los mutantes no equivalentes generados por dicho operador. Si M es el conjunto de mutantes generados por un programa p, T es el conjunto de casos de prueba, E es el conjunto de mutantes equivalentes producidos por el programa p y K_m es el conjunto de casos de prueba que matan al mutante m, la efectividad viene dada por $\mathcal{E} = (\sum_{m \in M} |K_m|)/((|M| - |E|) \cdot |T|)$.

Smith y Williams [17] han propuesto una clasificación de los mutantes en varios tipos: *dead on arrival* (DOA), aquéllos que son matados por el conjunto de casos de prueba inicial, *Killed* (matados por un caso de prueba específico añadido al conjunto de casos de prueba inicial), *crossfire* (matados por un caso de prueba diseñado específicamente para matar a otro mutante diferente), y *Stubborn* (los que no han podido ser matados añadiendo casos de prueba adicionales). Definen la función de utilidad de un operador como una combinación lineal de los porcentajes de los mutantes de cada tipo que genera el operador: $\mu = \alpha \cdot Killed + \beta \cdot DOA + \beta \cdot Crossfire - \alpha \cdot Stubborn$

Un inconveniente de este enfoque es que la definición de utilidad no sólo depende del conjunto de casos de prueba inicial, sino también del orden en que son escogidos los mutantes supervivientes después de su ejecución frente al conjunto de casos de prueba inicial. Además, no es fácil justificar la elección de los valores para los parámetros α and β en la definición.

Estero y col. [8] han evaluado cuantitativamente los operadores de mutación de WS-BPEL 2.0. Para ello, han medido la efectividad de conjuntos de casos de prueba adecuados y no redundantes, calculando el porcentaje de mutantes equivalentes, no válidos, resistentes y débiles que produce cada operador.

El trabajo que se presenta propone un enfoque alternativo a los anteriores para medir la calidad de los operadores de mutación. En primer lugar se define la calidad de un mutante individual y a partir de esta se define la calidad de un conjunto de mutantes, dicho conjunto puede ser el formado por los mutantes generados por un operador de mutación. A diferencia de los trabajos de Derezińska [5] y Smith y Williams [17], en este se utilizan conjuntos de casos de prueba que cumplen dos propiedades fundamentales para poder hacer estudios comparativos, los conjuntos de casos de prueba son adecuados y no redundantes.

8. Conclusiones y Trabajo Futuro

Este trabajo presenta una métrica de la calidad de los mutantes que se generan en la prueba de mutaciones, así como una métrica de la calidad de los operadores de mutación, basada en la anterior.

La métrica de calidad de los mutantes se ha aplicado a cuatro composiciones WS-BPEL, obteniéndose los porcentajes de mutantes de calidad que generan considerando cuatro umbrales diferentes de calidad. Los resultados obtenidos muestran que para composiciones que generan pocos mutantes se puede utilizar un umbral de calidad de $1-1/2\,|T|$, mientras que para composiciones que generan un número mayor el valor umbral puede ser $1 - 1/4\,|T|$.

También se ha aplicado la métrica de calidad de los operadores a tres composiciones WS-BPEL, midiéndose la calidad de 19 de sus 26 operadores. De los 19 operadores estudiados, 13 son considerados de alta calidad. Otros tres operadores, EMD, EMF y XMF, requieren una revisión de su implementación para disminuir el número de mutantes equivalentes que producen. Por último, ASF y APM han sido considerados de baja calidad, por lo que requiren un estudio posterior para definitivamente descartarlos.

Como trabajo futuro, planteamos ampliar el estudio a un mayor número de composiciones que permitan ejercitar todos los operadores de WS-BPEL. Por otro lado, dada la dependencia entre la definición de calidad de un mutante con el conjunto de casos de prueba utilizado, planteamos realizar un estudio estadístico con diferentes conjuntos de casos de prueba para ver si esta dependencia es significativa. Además, queremos estudiar la efectividad de las técnicas de reducción de mutantes en función de la calidad del subconjunto de mutantes que generan, y así establecer su grado de eficacia.

Referencias

1. Acree, A.T.: On Mutation. Ph.D. thesis, Georgia Institute of Technology (1980)
2. Ammann, P., Offutt, J.: Introduction to Software Testing. Cambridge University Press (2008)
3. Boubeta-Puig, J., Medina-Bulo, I., García-Domínguez, A.: Analogies and differences between mutation operators for WS-BPEL 2.0 and other languages (2011)
4. Budd, T.A.: Mutation Analysis of Program Test Data. Ph.D. thesis, Yale University (1980)
5. Derezińska, A.: Quality assessment of mutation operators dedicated for C# programs. In: QSIC 2006: Sixth International Conference on Quality Software. pp. 227–234. IEEE Computer Society Press (2006)
6. Domínguez-Jiménez, J., Estero-Botaro, A., García-Domínguez, A., Medina-Bulo, I.: Evolutionary mutation testing http://dx.doi.org/10.1016/j.infsof.2011.03.008. Information and Software Technology (2011)
7. Estero-Botaro, A., Palomo-Lozano, F., Medina-Bulo, I.: Mutation operators for WS-BPEL 2.0. In: ICSSEA 2008, 21th International Conference on Software & Systems Engineering and their Applications (2008)
8. Estero-Botaro, A., Palomo-Lozano, F., Medina-Bulo, I.: Quantitative evaluation of mutation operators for WS-BPEL compositions. In: IEEE International Conference on Software Testing Verification and Validation Workshops. pp. 142–150. IEEE Computer Society (2010)
9. Hussain, S.: Mutation Clustering. Master's thesis, King's College London (2008)
10. Jia, Y., Harman, M.: Higher order mutation testing. Information and Software Technology 51(10), 1379–1393 (2009)
11. MuBPEL: https://neptuno.uca.es/redmine/projects/sources-fm/wiki/MuBPEL
12. NetBeans Enterprise: `http://www.netbeans.org/kb/55/understand_trs.html`
13. OASIS: Web Services Business Process Execution Language 2.0. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html (2007)
14. Offutt, A.J., Untch, R.H.: Mutation Testing for the New Century, chap. Mutation 2000: Uniting the Orthogonal, pp. 34–44. Kluwer Academic Publishers (2001)
15. Offutt, A.J., Lee, A., Rothermel, G., Untch, R.H., Zapf, C.: An experimental determination of sufficient mutant operators. ACM Transactions on Software Engineering and Methodology 5, 99–118 (1996)
16. Polo Usaola, M., Reales Mateo, P.: Mutation testing cost reduction techniques: A survey. Software, IEEE 27(3), 80–86 (2010)
17. Smith, B.H., Williams, L.: On guiding the augmentation of an automated test suite via mutation analysis. Empirical Software Engineering 14(3), 341–369 (2009)