Analyzer4BPEL: Una herramienta de análisis de composiciones WS-BPEL para su aplicación en la etapa de prueba del software

Kevin J. Valle-Gómez, M. Carmen de Castro-Cabrera, Inmaculada Medina-Bulo

Departamento de Ingeniería Informática, Universidad de Cádiz, España,
kevin.vallegomez@alum.uca.es
{maricarmen.decastro, inmaculada.medina}@uca.es

Resumen Toda herramienta de prueba de software requiere en algún momento de su proceso realizar un análisis, o bien como primer paso, para extraer la información necesaria para aplicar una determinada técnica, o bien, una vez procesado el software para estudiar los resultados y evaluarlos. Existen en la actualidad herramientas capaces de analizar la mayoría de los lenguajes de programación. Sin embargo, no abundan los analizadores de lenguajes para composiciones de servicios. En este trabajo se presenta una aplicación que realiza un análisis de composiciones en lenguaje WS-BPEL y de sus casos de prueba para extraer información útil para diversos objetivos. Así mismo, se describe su utilización particular, en una de las etapas de la aplicación de una técnica de prueba de software: la prueba metamórfica.

Keywords: análisis, WS-BPEL, BPTS, prueba del software, prueba metamórfica, relaciones metamórficas, casos de prueba, servicios web

1. Introducción

Siempre que se desea trabajar con un problema es necesario analizarlo previamente, no importa su campo de aplicación, ya sea en las Matemáticas, la Física o la Arquitectura; se necesita obtener conocimiento previo a la solución de un problema. Este hecho no resulta diferente cuando se habla de software, donde antes de empezar a trabajar con él o se necesita comprender su funcionamiento, se deben estudiar además del contexto, todos los elementos del lenguaje de programación en el que esté implementado. La manera en la que está construído el programa, cómo se utilizan los elementos del lenguaje que se implementan o cuál es su propósito general.

Este análisis resulta fundamental cuando se quiere realizar cualquier proceso sobre el software, independientemente de su lenguaje, ya sea porque se necesita comprobar algo (mediante técnicas de prueba), porque se necesita modificar alguna parte con algún propósito, como la aplicación de los operadores de mutación de la técnica de prueba de mutaciones [8] o la creación de los mutantes

propios de esta técnica, o bien porque se necesita añadir algo, como hace la herramienta Takuan [12] que transformar ligeramente el software para la obtención de invariantes.
El lenguaje WS-BPEL [11] es un lenguaje específicamente diseñado para la implementación de procesos de negocio a partir de un conjunto de servicios web ya existentes. El resultado de la implementación de estos procesos de negocio consiste en una composición que actúa como servicio web independiente, siendo posible su reutilización en otras composiciones WS-BPEL. La herramienta *Analyzer4BPEL* está diseñada para trabajar específicamente con los elementos de este lenguaje.
Un proceso WS-BPEL está formado por *actividades* que pueden ser de dos tipos: *básicas y estructuradas*. Mientras que las *básicas* tienen un propósito único (obtener un dato, enviar un mensaje, ...), las *estructuradas* se encargan de definir la lógica del proceso de negocio y pueden estar compuestas a su vez de otras actividades, siendo a su vez también básicas o estructuradas.
En este trabajo se presenta *Analyzer4BPEL*, una herramienta capaz de analizar y extraer información de las composiciones de servicios WS-BPEL [11] y sus casos de prueba, así como, un ejemplo de aplicación en el análisis previo requerido en la Prueba Metamórfica (PM) [6], donde es especialmente necesario.

Este trabajo comienza con una descripción de la herramienta *Analyzer4BPEL* en la Sección 2, seguido de la descripción de la mejora en la aplicación para el reconocimiento de plantillas en los casos de prueba en la Sección 3. Finalmente, se muestra su aplicación a un caso de uso en la Sección 4 terminando con unas conclusiones en la Sección 5.

2. Analyzer4BPEL: Descripción de su funcionamiento

Se ha implementado una herramienta tomando como referencia el analizador requerido en el proceso de análisis y obtención de Relaciones Metamórficas (RM, en adelante) de la arquitectura anteriormente presentada [3], resultando ser un proceso de extracción eficiente y eficaz de información contenida en las composiciones WS-BPEL. Como novedad, la actual aplicación es capaz de analizar la información tanto de los ficheros de pruebas BPTS (BPEL Test Suit) en bruto, como de los BPTS basados en plantillas, tanto csv, como velocity. Esta herramienta ha sido desarrollada utilizando el lenguaje de programación Java [2], tratando de reutilizar métodos incluídos en bibliotecas ya desarrolladas, especialmente aquellos relacionados con la extracción y modificación de actividades WS-BPEL de *test-generator-autoseed* [10]. El resultado principal de la aplicación será un informe que contiene, de una manera legible y tratable por otras herramientas, información acerca de los elementos de la composición analizada que son más susceptibles a ser utilizados por las técnicas de prueba del software, como la PM. Estos elementos forman un conjunto de variables, constantes, operaciones y todos aquellos elementos que aparecen en la composición y en los casos de prueba ya implementados. Este proceso se realiza mediante los siguientes pasos:

1. Extracción de elementos de las composiciones WS-BPEL. Todos los elementos de la composición son extraídos: actividades junto con sus atributos, variables y constantes.
2. Extracción de elementos de la suite de casos de pruebas BPTS : todos los elementos de los casos de prueba son extraídos, independientemente de cómo se implementen internamente, ya que podemos encontrarnos ficheros escritos en bruto o que utilizan plantillas CSV o Velocity.
3. Información cruzada: Toda la información obtenida y cargada en los pasos anteriores se cruza. En este paso se buscan además todos los elementos que potencialmente pueden ser utilizados en las técnicas de prueba del software, como las operaciones, así como, aquellos elementos presentes en la composición y los casos de prueba al mismo tiempo.
4. Generación de informes. Todos los resultados se presentan en un informe con diferentes formatos. No solo se guarda la información obtenida en el cruce final, sino que también se provee de los elementos extraídos en la composición y en los casos de prueba, ya que siguen siendo útiles como una versión más simple y legible de estos, dando la oportunidad de ser tratados por cualquier herramienta o persona que lo requiera.

La Figura 1 representa un esquema visual de la herramienta que implementa este proceso. Hay que tener en cuenta que con los informes nos referimos tanto a los primeros informes (obtenidos de los ficheros WS-BPEL y BPTS) sin ningún tratamiento como al informe final (obtenido del filtrado y cruzado).

2.1. Extracción de información

La composición WS-BPEL y sus casos de pruebas BPTS son recorridos mediante una búsqueda en preorden en la cual se busca extraer todo tipo de información que pueda resultar útil, como puede ser el análisis previo a la obtención de RM.
Para trabajar con los ficheros BPEL, resulta esencial un conjunto de métodos capaces de acceder al contenido del fichero y trabajar con todos los elementos del mismo. La herramienta *test-generator-autoseed* [10] implementa una serie de métodos capaces de explorar los ficheros BPEL, junto con unos métodos *getters* y *setters* capaces de explorar las actividades del lenguaje. Por este motivo se incorporan los módulos correspondientes, ya que resultan ideales en nuestra herramienta, evitando así la reimplementación de métodos en los que se ha trabajado anteriormente.

Esta información extraída, también llamada información útil, se define como un conjunto de datos más susceptibles a contener errores. Tras estudiar diferentes composiciones [13], se ha visto que algunas actividades definidas en el lenguaje WS-BPEL, tales como las encargadas de la gestión de eventos o de la interacción con servicios externos, no suelen ser utilizadas en la mayoría de procesos, ya que no es habitual que su corrección dependa del desarrollador de la composición. Esto no significa que no puedan utilizarse en la aplicación de

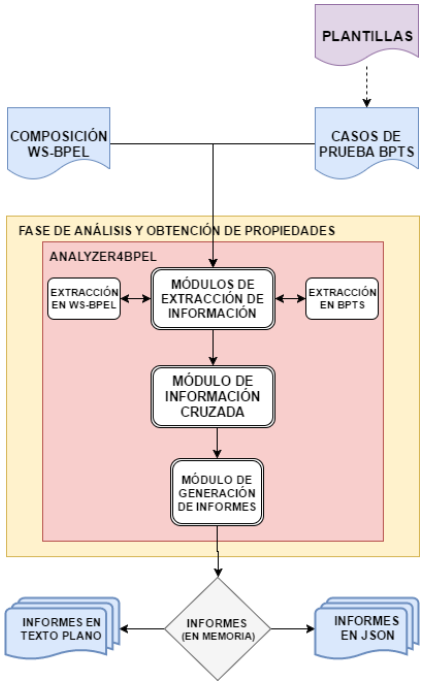


Figura 1: Esquema de la herramienta con todos sus pasos

técnicas de prueba, sino que tienen una menor probabilidad de ser utilizadas en comparación con otro tipo de actividades, por lo que se ha decidido mantener un informe con toda la información extraída de los ficheros sin filtrar junto al informe que representa el resultado final.
En los casos de prueba BPTS, lo que se proporciona es un informe con un formato común que contiene toda la información de los diferentes casos de prueba, el cual resulta útil en técnicas donde se generan otros casos de prueba a partir de los iniciales.

2.2. Información cruzada

Una vez que se extrae toda la información, esta se envía al módulo de generación de informes para generar los dos informes iniciales sin filtrar: uno para la composición WS-BPEL y otro para los casos de prueba BPTS. Sin embargo, esta información se mantiene y se envía al módulo de información cruzada.

En este paso, se buscan elementos coincidentes entre los datos extraídos de los distintos elementos. Aquellos que aparecen tanto en la composición como en el caso de prueba se guardan aparte, así mismo, se guardan aquellas actividades que consideramos que generan información útil, como las variables, constantes

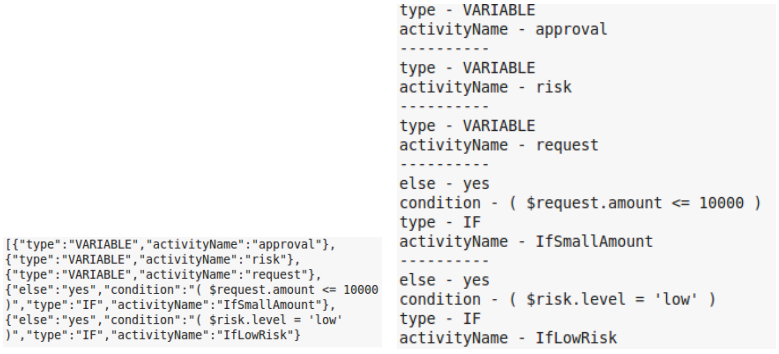
y algunas operaciones. Finalmente, se envía toda la información al módulo de generación de informes para generar un informe final.

2.3. Representación del informe

Cuando se presentó la idea inicial [4], se consideró el uso del formato CSV [7] para la generación de un informe con un formato más entendible por otras herramientas de cara a permitir la automatización del manejo de la información obtenida por el proceso implementado. Después de la implementación del primer prototipo, se siguió estudiando la posibilidad de utilizar otros formatos, descubriendo que para este propósito el formato JSON resultaba mejor que CSV.

El formato JSON [9] provee de una sintaxis para almacenar e intercambiar datos utilizando la notación de JavaScript. Este formato es muy flexible y puede ser utilizado por prácticamente cualquier lenguaje de programación actual, haciendo de su interpretación un proceso realmente sencillo. Con este formato se puede presentar un informe con la información precisa sin necesidad de utilizar elementos externos.

Aunque JSON no es un formato realmente difícil de leer y existen multitud de intérpretes que facilitan su lectura manual, existe la necesidad de un informe que pueda ayudar a cualquier persona en la tarea del análisis manual de tener que acudir a elementos externos, como los intérpretes anteriormente mencionados. Es por eso por lo que además del informe en formato JSON, se genera un informe en texto plano con un formato genérico y entendible en el que se representan todas las etiquetas de los elementos junto con sus valores, sin ningún tipo de estructura visual que dificulte la lectura. En la Figura 2 podemos ver un fragmento de informe en JSON junto con su versión formateada.



(a) JSON (b) Texto formateado

Figura 2: Fragmento de informe en dos formatos: a)JSON y b)texto formateado

3. Uso de plantillas en los casos de prueba BPTS

Uno de los problemas más comunes encontrados en la aplicación de este proceso era el uso de plantillas que muchos casos de prueba BPTS utilizan para la especificación de sus casos de prueba. El uso de plantillas para este propósito tiene muchas ventajas y se está extendiendo cada vez más entre las composiciones implementadas hoy en día.
Hemos identificado dos formatos que se utilizan repetidamente entre todas las composiciones estudiadas [13]; CSV y Velocity. En la primera versión [4], solo se consideraban los casos de prueba BPTS cuyos datos estaban directamente escritos en el mismo fichero. Esto resultaba suficiente a la hora de comprobar que el proceso se aplicaba correctamente, pero no resulta suficiente para su uso en un entorno real. Este hecho crea la necesidad de añadir compatibilidad con el uso de plantillas en los casos de prueba de manera que no se altere el proceso y resultando, en la medida de lo posible, transparente para el usuario.
Se ha mejorado el proceso considerando el uso de las diferentes plantillas y quedando preparado de manera que cualquier formato emergente pueda ser fácilemnte adaptado en el futuro. Se han considerado los dos formatos principales mencionados anteriormente. Aunque el proceso resulta transparente para el usuario y no hay diferencia de resultados en los informes, la manera en la que los datos son tratados varía internamente.
Hay que tener en cuenta que el formato de las plantillas utilizadas en la implementación de los casos de prueba es independiente del formato elegido para la representación de los informes.

Si se observan los ficheros BPTS que cargan sus valores desde una plantilla, se encontrarán algunos elementos interesantes. En primer lugar, como podemos ver en los elementos del Listado 1.1, encontraremos un elemento *'dataSource'* que contiene el tipo de plantilla y su localización.
Podemos observar además en el Listado 1.1 que existen algunas diferencias cuando se trata de una plantilla CSV y de una plantilla Velocity. En el caso de la plantilla CSV, se indicará el separador que se utiliza en este fichero, que en este caso sería la coma. En el caso de la plantilla Velocity, se muestran las variables que irán iterando para su asignación de valores.

```
<tes:dataSource type="csv" src="data.csv">
    <tes:property name="separator">,</tes:property>
</tes:dataSource>

<tes:dataSource type="velocity" src="data.vm">
  <tes:property name="iteratedVars">
        req_amount ap_reply ap_limit as_reply as_limit accepted
  </tes:property>
</tes:dataSource>
```

Listing 1.1: Propiedad dataSource (fragmento de fichero BPTS)

En este caso, la plantilla se encontrará en el mismo directorio que el fichero BPTS, por lo que no será necesario especificar más que el nombre del fichero.

El resto del fichero BPTS implementa la funcionalidad de los casos de prueba exactamente igual que cuando se implementa con los datos en bruto, solo que utilizando las variables definidas por las plantillas cuando sea necesario. Aunque la funcionalidad de los casos de prueba no es necesaria en este ejemplo, en el caso de la composicion del triángulo, cuyo tratamiento se muestra con detalle en el caso de uso descrito más adelante, encontraríamos el nombre de la variable junto con el símbolo '$', es decir, *$TA, $TB* y *$TC*.

Se ha adaptado la herramienta para reconocer de manera automática las plantillas y generar los informes con los datos que se obtienen de las mismas.
Cuando se detecta la fuente del elemento *'dataSource'*, automáticamente se buscará el fichero especificado en el atributo 'src'. Si este no existe, significa que se ha encontrado un error claro en la implementación de los casos de prueba y se notificará de manera correspondiente, generando un único informe de extracción del fichero WS-BPEL. Si se encuentra, se cargarán los valores y se obtendrán los casos de prueba mediante la iteración de los valores con su plantilla, obteniendo así el número necesario de casos de prueba.

Finalmente, una vez obtenidos los casos de prueba, se continuará con el cruce de información correspondiente, generando unos informes con el mismo formato implementado anteriormente. De cara al usuario, este proceso no tiene ninguna diferencia, puesto que el resultado y el tiempo de ejecución son similares, pero se consigue la ventaja de abstracción de la implementación hasta un cierto límite para el propósito final.

4. Aplicación a un caso de uso

Como se ha mencionado anteriormente, la implementación de la fase de análisis y obtención de propiedades en la aplicación de la PM exige el uso de un buen analizador. Se ha aplicado esta herramienta para este propósito, por lo que a continuación se describe la técnica y un ejemplo sobre cómo se puede obtener una RM utilizando solo el informe generado por la herramienta.

4.1. Prueba Metamórfica

El problema del oráculo es una de las mayores preocupaciones en todas las técnicas de prueba que se utilizan actualmente. Llamamos oráculo a aquella entidad capaz de determinar si el comportamiento del software que se está probando es el correcto. El oráculo puede ser una persona con conocimiento suficiente acerca del problema, así como otro software que compruebe el funcionamiento [14].
El problema del oráculo consiste en la ausencia del mismo, es decir, hay casos en los que no se puede comprobar que el comportamiento es el correcto, ya sea

porque la persona encargada cometa algún error o porque el software es tan complejo que demostrar su correcto funcionamiento tiene un coste demasiado elevado, mayor aun que el de ejecutar el propio software. El uso de la PM ha sido elogiado en su aplicación para aliviar este problema [5] [6].
El concepto de PM está muy relacionado con el de RM. Una de las definiciones de RM [1] dice que son propiedades existentes que relacionan la entrada de un programa con una salida esperada, obtenida tras múltiples evaluaciones de una función; así que podemos decir que cuando la entrada es correcta, se esperará siempre un resultado que cumpla con las propiedades específicadas. Las RM representan una manera eficaz de generar nuevos casos de prueba a partir de otros ya existentes.

4.2. Ejemplo de relación metamórfica

Vamos a describir el concepto de RM mediante un ejemplo. Se considera una composición que determina la geometría de un triángulo dados sus lados: isósceles, equilátero o escaleno. Si se le proporciona un triángulo con tres lados iguales *(a, a, a)*, sabemos con certeza que la composición debe devolver siempre el resultado 'equilátero'. Esto es una propiedad geométrica que debe cumplirse siempre sin excepción, por lo que si obtenemos cualquier otro resultado, es seguro que habrá algún error.

Un triángulo equilátero mantendrá su geometría siempre y cuando se respete la proporcionalidad de sus lados, por lo que si multiplicamos los lados por un mismo número natural $(a \cdot n, a \cdot n, a \cdot n)$, el resultado será siempre el mismo. Este hecho es una RM expresada en lenguaje natural, a continuación la expresamos formalmente:

$$RM_1 : \exists T_1, T_1', R_1, R_1' \ tal\ que\ \forall n \epsilon \mathbb{N} > 0 \Rightarrow T_1' = n \cdot T_1 \wedge R_1 = triangle(T_1) \wedge R_1 = \text{equilátero} \wedge R_1' = triangle(T_1') \Rightarrow R_1' = \text{equilátero}$$

4.3. Ejemplo de aplicación: la composición del triángulo

En esta sección se comprobará el resultado obtenido tras la extracción de datos de la composición del Triángulo anteriormente propuesta en la Sección 4.2, así cómo un ejemplo de como podría esta información llevar a la obtención de RM. El propósito de este ejemplo es mostrar que es posible obtener RM utilizando solo la información contenida en los informes y el conocimiento sobre el dominio del problema.

1. Proveemos a la aplicación con la composición WS-BPEL y el fichero BPTS, lo que generará tres informes. Buscaremos RM utilizando solo estos informes.
2. Observando el informe final encontramos tres variables: *ta*, *tb* y *tc*, que representan los lados del triángulo. Este informe también nos muestra que hay varias condiciones que se evalúan durante la condición.

3. Cogeremos solo la primera condición para este ejemplo, que comprueba si es un triángulo válido.
   $cond0 = (ta > 0 \wedge tb > 0 \wedge tc > 0)$
4. Con los conocimientos básicos de geometría de triángulos, sabemos que si esta condición no se cumple, la figura no será un triángulo, por lo que el resultado obligatoriamente será un *triángulo inválido*.
5. Teniendo en cuenta la lógica de la composición, sabemos que si otorgamos el mismo valor a dos lados del triángulo y otro valor diferente al tercero, todos los valores son mayores que 0 y cumple las condiciones de formación de triángulos, podríamos obtener un triángulo *isósceles*.
6. Mediante los hechos de los pasos 4 y 5, podemos deducir que si a uno de los lados de dicho triángulo le restamos un valor mayor o igual a sí mismo, obligatoriamente obtendremos como resultado un *triángulo inválido*. Si esto no se cumple, sabemos que hay algún problema, por lo que podemos considerar este hecho como RM, la cual es representada a continuación en un lenguaje formal.

$$RM_1 : \exists T_1, T_1', R_1, R_1' \ tal\ que\ \forall n, p \epsilon \mathbb{N} > 0 \wedge \forall m \epsilon \mathbb{N} \geq n \Rightarrow T_1 = (n, p, p) \wedge R_1 = \text{isósceles} \wedge T_1' = (n - m, p, p) \Rightarrow R_1' = \text{triángulo inválido}$$

Por tanto, si en la composición hay un error y en la condición $cond0 = (ta > 0 \wedge tb > 0 \wedge tc > 0)$ se ha escrito el operador lógico $\vee$ en lugar de $\wedge$ en su primera ocurrencia, el ejemplo del paso anterior podría dar un resultado diferente al esperado, al no detectarse que uno de los lados tiene un valor menor o igual a cero, por lo que no se cumpliría la RM_1 y se detectaría un error en la composición.

5. Conclusiones y Trabajo Futuro

En este trabajo se ha presentado *Analyzer4BPEL*, una herramienta que analiza y extrae información de las composiciones WS-BPEL y sus casos de prueba BPTS, junto con su aplicación a la PM para el análisis previo a la obtención de RM en la fase de análisis y obtención de propiedades descrita en la arquitectura anteriormente presentada [3].

Esta herramienta ha sido adaptada para trabajar con casos de prueba BPTS que implementen casos de prueba escritos en bruto, que utilicen plantillas CSV y que utilicen plantillas Velocity, aunque se ha preparado para una posible ampliación respondiendo a las necesidades de uso que puedan surgir en los diferentes entornos de aplun icación, todo ello de una manera totalmente transparente para el usuario final. La intención es que sea una aplicación de software libre para dar la posibilidad de ser utilizada para otro software en que se requiera.

La información obtenida por la aplicación se utilizará en la fase de generación de RM, como parte de un framework que aplica PM a composiciones de servicios. Se seguirá estudiando la aplicación del análisis implementado por esta herramienta para otros propósitos. Así mismo, se estudiará su impacto a la hora de comprender el funcionamiento de las distintas composiciones WS-BPEL.

6. Agradecimientos

Este trabajo ha sido financiado por el Programa Nacional para la Investigación, Desarrollo e Innovación del MICINN y fondos FEDER, proyecto DArDOS (TIN2015-65845-C3-3-R).

Referencias

1. J. H. Andrews, L. C. Briand, and Y. Labiche. Is mutation an appropriate tool for testing experiments? In *Proceedings of the 27th International Conference on Software Engineering (ICSE 2005)*, pages 402–411. ACM Press, 2005.
2. Ken Arnold, James Gosling, and David Holmes. *The Java programming language*. Addison Wesley Professional, 2005.
3. M. del Carmen Castro-Cabrera and Inmaculada Medina-Bulo. Análisis y especificación de propiedades para la prueba metamórfica en WS-BPEL. In *Actas de las IX Jornadas de Ciencia e Ingeniería de Servicios*, pages 155–162, Madrid, Spain, September 2013.
4. M. del Carmen Castro-Cabrera, Kevin J. Valle-Gómez, and Inmaculada Medina-Bulo. Automatización de la etapa de análisis para la aplicación de la técnica de prueba metamórfica a composiciones de servicios WS-BPEL. In *Actas de las XII Jornadas de Ciencia e Ingeniería de Servicios*, Salamanca, Spain, 2016.
5. T. Y. Chen. Metamorphic testing: A new approach for generating next test cases. *HKUSTCS98-01*, 1998.
6. T. Y. Chen. Metamorphic testing: A simple approach to alleviate the oracle problem. In *Proceedings of the 5th IEEE International Symposium on Service Oriented System Engineering*. IEEE Computer Society, 2010.
7. Edoceo. Comma separated values (csv) standard file format. http://edoceo.com/utilitas/csv-file-format, 2014 (accessed Dec 30, 2016).
8. Yue Jia and Mark Harman. An analysis and survey of the development of mutation testing. *Software Engineering, IEEE Transactions on*, 37(5):649–678, 2011.
9. JSON.org. Json. http://www.json.org, (accessed Dec 26, 2016).
10. Valentín Liñeiro Barea. Herramienta para la generación automática de casos de prueba mediante siembra automática para WS-BPEL 2.0. (April 2016), 2014.
11. OASIS. Web Services Business Process Execution Language 2.0. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html, 2007. Organization for the Advancement of Structured Information Standards.
12. Manuel Palomo Duarte, Alejandro Álvarez Ayllón, Antonio García Domínguez, and Inmaculada Medina Bulo. La cobertura de los casos de prueba en la generación dinámica de invariantes en composiciones WS-BPEL. *Actas de Talleres de Ingeniería del Software y Bases de Datos, San Sebastián*, 2009.
13. UCASE Research Group. *WS-BPEL Composition Repository - Redmine*, (accessed Dec 30, 2016).
14. Elaine Weyuker. On testing Non-Testable programs. *The Computer Journal*, 25(4):465–470, November 1982.