Escuela Superior de Ingeniería

Grado en Ingeniería Informática

Reingeniería de Software para Análisis Armónico de las Mareas

Curso 2018-2019

Arantzazu Otal Alberro

Puerto Real, 3 de abril de 2019

Escuela Superior de Ingeniería

Grado en Ingeniería Informática

Reingeniería de Software para Análisis Armónico de Mareas

Departamento: Ingeniería Informática.
Directora del proyecto: Mª del Carmen de Castro Cabrera
Autor del Proyecto: Arantzazu Otal Alberro.

Puerto Real, 3 de abril de 2019

Fdo.: Arantzazu Otal Alberro

Agradecimientos a mi tutora Mª del Carmen de Castro Cabrera, por todo el apoyo, la dedicación y el trabajo realizado. A todos los profesores del grado que me han aportado los suficientes conocimientos para lograr mis objetivos en el ámbito académico y laboral. A mis padres por confiar en mí e inculcarme valores sobre la importancia del saber y de superarme cada día. A mis compañeros, por compartir las dificultades y los retos que nos ha supuesto la carrera y por la ayuda que me han proporcionado. A mi pareja, por darme ánimos todos los días y recordarme las cualidades buenas que tengo y además resolverme dudas ya que compartimos la misma pasión por dedicarnos a esta profesión.

A todas las personas que de alguna u otra manera me han ayudado y apoyado, haciendo que este trabajo haya sido posible.

Gracias.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

1.1. Motivación

Existen multitud de técnicas para probar el software que se desarrolla diariamente, así mismo, estas suelen ser relativamente sencillas de aplicar. Sin embargo, en muchas ocasiones no se realizan todas las pruebas necesarias, ya que las personas encargadas de realizarlas asumen las limitaciones que ofrecen las técnicas de prueba [25], provocando la existencia de errores difíciles de detectar en prácticamente todos los sistemas existentes.

En [7] se presentó una arquitectura para aplicar la técnica de prueba metamórfica en WS-BPEL. Esta arquitectura pretende mejorar el conjunto de casos de prueba de las composiciones WS-BPEL aplicando las relaciones metamórficas. De esta manera, se generarán nuevos casos de prueba que detectarán errores en las composiciones que no se detectaban con el conjunto de casos de prueba inicial.

En la Figura 1.1 pueden verse todas las etapas de la arquitectura propuesta.

Este Trabajo de Fin de Grado se centra en la *etapa de análisis y obtención de propiedades*, donde se trata de analizar la composición y sus componentes para obtener propiedades importantes para la especificación de las relaciones metamórficas.

Resulta importante la existencia de un analizador eficiente que sea capaz de obtener información útil de la manera más rápida posible. Aunque el uso de la herramienta Takuan [17] ha demostrado un buen funcionamiento ofreciendo unos buenos resultados, resulta necesario probar otras herramientas o técnicas de análisis para esta etapa.

La motivación principal de este Trabajo de Fin de Grado consiste en desarrollar una herramienta de análisis y obtención de información que se adapte por completo a las necesidades de esta arquitectura. Así mismo, esta herramienta sería utilizada para obtener información de las composiciones WS-BPEL y sus casos de prueba con el fin de ayudar a obtener las relaciones metamórficas. Con un formato adecuado, esta herramienta será de utilidad para la aplicación de otras técnicas de prueba, así

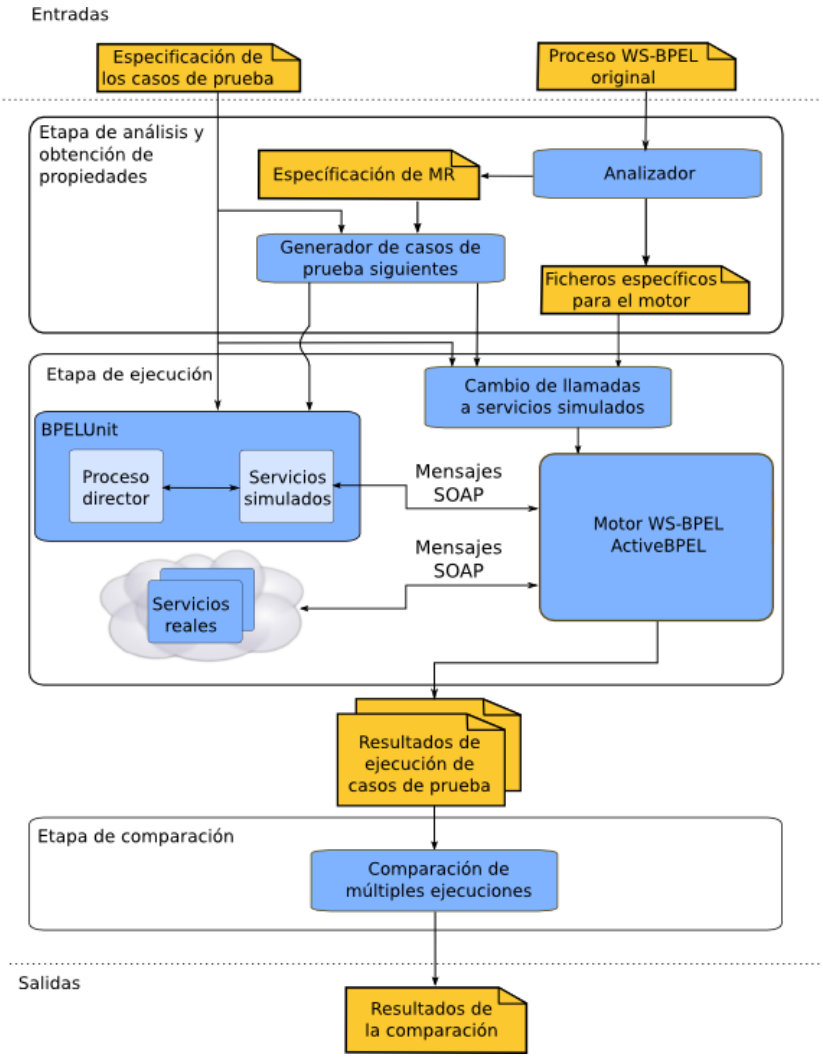

Figura 1.1.: Arquitectura para aplicar la prueba metamórfica en WS-BPEL

como del análisis puntual que deba hacerse sobre una composición.

1.2. Objetivos

El objetivo principal de este Trabajo de Fin de Grado consiste en el desarrollo de una herramienta que analice de y extraiga información relevante de las composiciones en WS-BPEL y los casos de prueba implementados para las mismas para facilitar la obtención y elección de relaciones metamórficas. Esta información se presentará en dos formatos, con el objetivo de que otras herramientas sean capaces de leerla

con la menor dificultad, pero que de igual manera una persona sea capaz de leer un informe detallado acerca de los elementos que debe tener en cuenta para obtener relaciones metamórficas en ese contexto.
Para alcanzar este objetivo será necesario realizar los siguientes pasos:

- Análisis de la composición y sus casos de prueba: Se explora y se extrae el contenido de la composición WS-BPEL y sus casos de prueba.
- Generación de informe detallado: Se genera un informe detallado con información del contenido de la composición y sus casos de prueba en los dos formatos propuestos.
- Análisis de la información obtenida: Se explora el contenido de los informes obtenidos en el análisis de la composición y sus casos de prueba con el objetivo de encontrar aquellos elementos más relevantes en el conjunto de ambos.
- Generación de informe final: Se genera un informe final con la información obtenida del segundo análisis. Este informe, presentado en los dos formatos propuestos, contiene la información más relevante a la hora de obtener relaciones metamórficas, pudiendo ser utilizada por otra herramienta para automatizar todo el proceso incluyendo la obtención de relaciones metamórficas o por la persona encargada de esta tarea.

1.3. Alcance

En este Trabajo de Fin de Grado se estudian y analizan las diferentes actividades propias del lenguaje WS-BPEL, centrándose principalmente en los elementos más comunes de estas. Así mismo, se desarrolla una herramienta capaz de generar información que puede ayudar al desarrollo y obtención de relaciones metamórficas a partir de una composición y un caso de prueba dados.

— Así mismo, este proyecto genera los siguientes productos:

- Estudio de las composiciones WS-BPEL, los casos de prueba BPTS y sus diferentes elementos para la toma de información considerada relevante.
- Herramienta que analiza por completo una composición WS-BPEL y sus casos de prueba, generando la salida correspondiente.
- Fichero *.json* que contiene toda la información relevante de las composiciones lista para su uso.
- Fichero *.txt* que contiene un informe acerca de la información relevante obtenida.

1.4. Lenguaje WS-BPEL

El lenguaje WS-BPEL 2.0 -siglas de *Web Services Business Process Execution Language*- es un lenguaje estandarizado por OASIS, específicamente diseñado para definir procesos de negocio a través de la especificación de servicios web. Está basado por completo en el lenguaje XML -siglas de *Extensible Markup Language*-, lo cual permite que sea independiente del motor donde esté siendo ejecutado.

Los procesos implementados en WS-BPEL tienen una estructura claramente diferenciada, dividida en secciones independientes, las cuales son:

1. 'Partner Links': En esta sección se definen la relación del proceso de negocio con los servicios externos implicados en el proceso.

2. 'Variables': En esta sección se definen los datos variables que se utilizan en el proceso. De estas variables se proporciona su definición en forma de tipos de mensaje WSDL, tipos XML Schema o elementos XML Schema. Estas variables permiten a los procesos de negocio mantener un estado entre el intercambio de mensajes.

3. 'Event Handlers': Manejadores de eventos. En esta sección se definen los distintos tipo de manejadores de eventos que pueden aparecer en el proceso. Los más comunes son los manejadores de fallo, que contienen la lógica necesaria para manejar los fallos producidos a lo largo del proceso, pero existen multitud de tipos, como los manejadores de evento, terminación o compensación.

4. 'Process': El resto del proceso de negocio consiste en la sección de proceso. En esta sección se encuentra la descripción del comportamiento normal que debe seguir el proceso de negocio, implementado mediante las actividades propias del lenguaje.

Las actividades proporcionadas por el lenguaje WS-BPEL permiten implementar una actividad específica dentro de un proceso de negocio. Normalmente aparecen recogidas en una actividad global llamada *'sequence'* o *'flow'*.

1.4.1. Actividades del lenguaje WS-BPEL

Las actividades del lenguaje WS-BPEL están representadas por un elemento XML, a los cuales se le pueden asociar una serie de atributos y un conjunto de contenedores, los cuales a su vez pueden tener lo mismo. Existen dos tipos de actividades:

- Actividades básicas: son aquellas que realizan una labor específica, como puede ser la emisión o recepción de un mensaje, la invocación de servicios, etc.

- Actividades estructuradas: son aquellas que pueden contener en su interior otras actividades básicas o estructuradas, y se encargan de definir toda la lógica de negocio.

En la Figura 1.2 podemos ver como sería una actividad estructurada de manera gráfica.

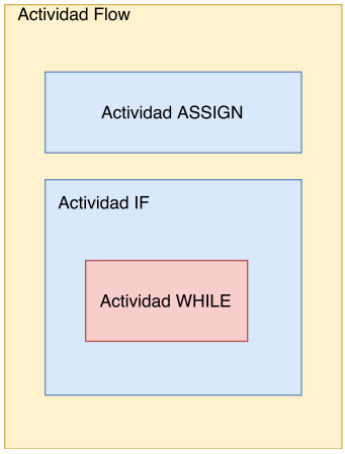

Figura 1.2.: Actividad estructurada en WS-BPEL

Como se menciona en la Sección 1.4, las actividades que componene el proceso aparecen recogidas en una o varias actividades globales, que pueden ser *'sequence'* o *'flow'*. El lenguaje WS-BPEL ofrece soporte nativo para la concurrencia, esto es posible utilizando la actividad *'flow'*. Esta actividad permite ejecutar un conjunto de actividades de forma concurrente, pudiendo especificar las condiciones de sincronización que deben producirse entre las mismas. De igual manera, podemos ejecutar un conjunto de actividades donde se ejecutarán en el mismo orden en el que aparecen, sin utilizar la concurrencia. Esto es posible utilizando la actividad *'sequence'*.

Aunque se se recoge la definición de una gran cantidad de actividades en la especificación lenguaje WS-BPEL, hay algunas más importantes y utilizadas que otras, las cuales pueden ser:

- 'Assign': Esta actividad provee un método destinado a la manipulación de datos. Mediante la actividad *assign* se puede transferir información entre variables, expresiones y otros elementos del proceso de negocio.

- 'If': Esta actividad permite definir un comportamiento condicional en casos específicos, donde se debe elegir entre distintas ramas de comportamiento. Solo se seguirá un camino dependiendo del resultado de la evaluación de la condición propuesta. Suelen ir acompañadas de actividades *else* o *elseif*.

- 'Else': Esta actividad complementa a la actividad condicional *If*, y es ejecutada siempre y cuando no se cumpla la condición especificada en la misma.
- 'ElseIf': Esta actividad actúa de manera similar a la actividad *Else*. La principal diferencia reside en la evaluación de una nueva condición. Si la condición no se cumple, no se seguirá por el camino establecido, pudiendo estar acompañada de nuevo por otra actividad *else* o *elseif*.
- 'ForEach': Esta actividad permite ejecutar un conjunto de actividades en secuencial o concurrentemente (se indica en un atributo). La actividad o conjunto de actividades presentes dentro de una actividad *ForEach* se ejecutará N+1 veces, siendo N el resultado de la resta del valor marcado como final (*finalCounterValue*) menos el valor marcado como inicial (*startingCounterValue*).
- 'Invoke': Esta actividad se encarga de llamar a otros servicios web en el momento indicado.
- 'Receive': Esta actividad de encarga de especificar toda la información recibida de los servicios externos.
- 'Reply': Esta actividad se encarga de enviar un mensaje de respuesta a la información recibida a través de otra actividad *Receive*.
- 'Scope': Esta actividad consiste en un conjunto de elementos en un ámbito local. Puede estar compuesta de otras variables, manejadores de eventos y actividades independientemente del ámbito global. Se podría decir que es análoga a los bloques { } de otros lenguajes de programación.
- 'While': Esta actividad permite ejecutar un conjunto de actividades varias veces de manera iterativa. La actividad o conjunto de actividades presentes dentro de una actividad *While* se ejecutará hasta que la condición evaluada no se cumpla, obteniendo el valor *false*.

1.4.2. Desafíos en la prueba de composiciones BPEL

De por sí, la fase de pruebas del software ya es una tarea que consume tiempo y dinero, lo que hace que en muchas ocasiones sea descuidada. [5]
Debido a la naturaleza de las composiciones WS-BPEL, la prueba de las mismas se hace mucho más complicada de lo que sería la prueba en otro tipo de software. Esto es debido a varios motivos.

- Rendimiento: Las composiciones WS-BPEL dependen del resultado de la llamada de multitud de servicios web externos. Las llamadas a servicios SOAP

tienen un coste muy alto, sumando todas las tareas que deben realizar los servicios web, hace que el rendimiento de las composiciones acabe siendo un lastre para la realización de las pruebas.

- Condiciones de los errores: Debido a la naturaleza distribuida de las composiciones WS-BPEL, hay que tener en cuenta una gran cantidad de errores que pueden darse debido al entorno de ejecución. Fallos de red o servicios externos que han dejado de funcoinar se encuentran entre los errores más comunes de este tipo.
- Dependencias: Como se ha mencionado anteriormente, las composiciones WS-BPEL tienen dependencias con servicios web externos. La composición debe asumir el correcto funcionamiento de estos servicios web para completar sus tareas. Sin embargo, en ocasiones existen pocos recursos que aseguren que el resultado de estas composiciones es el correcto.
- Despliegue: Las composiciones WS-BPEL deben ser lanzadas en un servidor específico, así como estar listas para su ejecución. Esto provoca que el tiempo transcurrido hasta que se puedan lanzar los test tienda a ser demasiado largo.
- Complejidad: Las composiciones WS-BPEL incluyen normalmente muchos elementos diferentes, como asignaciones, llamadas a servicios web externos o actividades propias del lenguaje. La variedad de situaciones posibles hace necesario que se deban tener en cuenta un número muy grande de casos de prueba.

1.4.3. Casos de prueba en composiciones WS-BPEL

Cada composición WS-BPEL se encuentra acompañado de uno o varios casos de prueba en formato BPTS. BPTS -*BPELUnit Test Suite*- es un formato utilizado para reunir uno o varios casos de prueba para la herramienta BPELUnit. Estos ficheros definen los procesos a probar y el procedimiento correspondiente en formato XML.

Este tipo de casos de prueba permite la utilización de plantillas a al hora de dar valores a los elementos que se desean probar, aumentando así la flexibilidad de los casos de prueba y proporcionando mayor sencillez a la hora de modificar estos valores, puesto que no será necesario modificar los ficheros BPTS para probar diferentes casos de prueba.
Podemos encontrar tres tipos:

1. Ficheros BPTS sin plantilla: Son aquellos casos de prueba donde el valor se encuentra en el propio fichero BPTS. No se hace uso de plantillas, por lo que no son necesarios los ficheros externos ni la exploración en busca de los valores.

2. Ficheros BPTS con plantilla CSV: Son aquellos casos de prueba donde el valor se encuentra en ficheros CSV. Mientras que en los casos de prueba aparece el nombre de la variable correspondiente donde deben aparecer los valores, los ficheros CSV tendrán como cabecera el nombre de esa variable, siendo la lista de valores todos los casos de prueba que se deben formar.

3. Ficheros BPTS con plantilla Velocity: Estos casos de prueba son similares a los que utilizan plantillas CSV, con la diferencia de que la plantilla aparece escrita en VTL [4]. Estas plantillas son mucho más flexibles y potentes, ya que aprovechan todas las ventajas del lenguaje proporcionado por Apache.

La herramienta desarrollada en este Trabajo de Fin de Grado se centra en los *Ficheros BPTS sin plantilla*, ya que son estos los que resultan más problemáticos a la hora de interpretar los elementos que se van a probar al no existir una lista de valores externa.

1.5. Composición del triángulo

Existe una composición WS-BPEL sobre la que se basarán la mayoría de ejemplos de este TFG. Así mismo, esta composición será el objetivo principal para la aplicación de los correspondientes casos de uso.

La composición *Triangle* -también conocida simplemente como triángulo- consiste en un proceso que a partir de los valores de los lados de un triángulo proporcionado es capaz de validar el mismo y determinar su geometría. Esta composición se implementa mediante el siguiente proceso:

1. Se proporciona la longitud de los tres lados del triángulo: a, b y c.

2. Se comprueba si los tres lados son mayor que 0
   $a > 0 \land b > 0 \land c > 0$
   a) Si todos son mayores que 0, continúa la ejecución.
   b) Si uno o varios lados menores o iguales a 0, termina la ejecución y **devuelve -1**, lo que significa que no se ha proporcionado un triángulo.

3. Se comprueba si los tres lados pueden formar un triángulo.
   $(a + b) > c \land (a + c) > b$
   a) Si la condición se cumple, continúa la ejecución.
   b) Si la condición no se cumple, termina la ejecución y **devuelve -2**, lo que significa que los lados proporcionados no pueden formar un triángulo.

4. Se comprueba si todos los lados son iguales.
   $a = b \wedge b = c$
   a) Si son iguales, termina la ejecución y **devuelve 3**, lo que significa que se trata de un triángulo equilátero.
   b) Si no son iguales, se pasará al paso 5.

5. Se comprueba si dos de sus lados son iguales.
   $a = b \vee b = c \vee a = c$
   a) Si la condición se cumple, termina la ejecución y **devuelve 2**, lo que significa que se trata de un triángulo isósceles.
   b) Si la condición no se cumple, termina la ejecución y **devuelve 1**, lo que significa que se trata de un triángulo escaleno.

De esta manera, podemos deducir una equivalencia entre los resultados y la geometría del triángulo proporcionado, representada en la Tabla 1.1

Valor	**Triángulo**
-2	Los lados no pueden formar un triángulo.
-1	No se ha proporcionado un triángulo.
1	El triángulo es escaleno.
2	El triángulo es isósceles.
3	El triángulo es equilátero.

Tabla 1.1.: Equivalencia de valores con el triángulo

Para ilustrar esta composición, puede verse el esquema BPMN en la Figura 1.3

1.6. JSON, JavaScript Object Notation

El formato JSON [13] -siglas de *JavaScript Object Notation*- es un formato ligero basado en un subconjunto del lenguaje de programación *JavaScript*, cuyo propósito es el intercambio de datos. Este formato resulta simple para ser escrito y leído por personas, mientras que para las máquinas ocurre lo mismo, su generación e interpretación resulta sencilla. JSON es un formato de texto completamente independiente del lenguaje con el que se utilice, aunque tiene ciertos elementos ampliamente comunes en los lenguajes de propósito general más usuales, como *C++, Java o Python*, lo que hace que JSON sea un lenguaje ampliamente usado en el intercambio de datos.

La información en formato JSON se compone de dos estructuras:

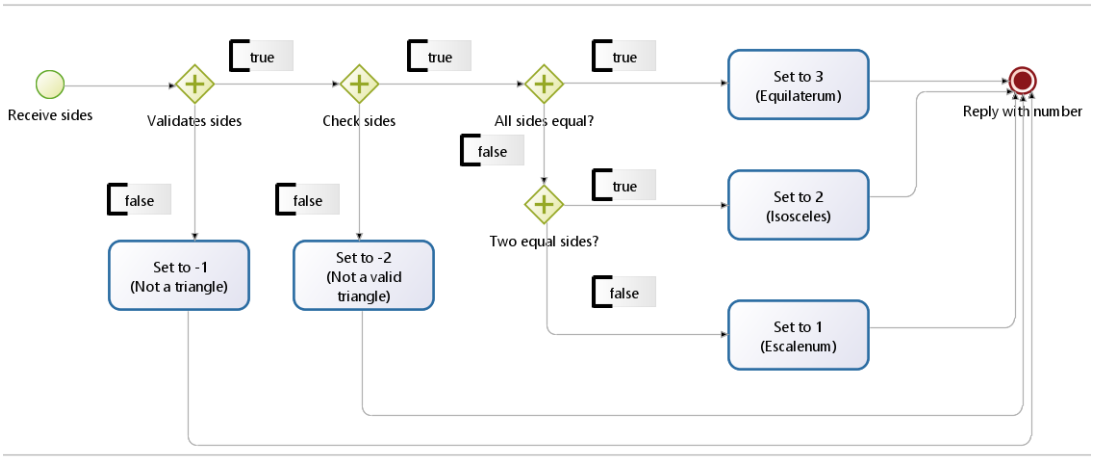

Figura 1.3.: Esquema BPMN de la composición Triangle

- Conjunto de pares nombre-valor: Se trata de una colección de datos en este formato, estando compuesto normalmente del nombre del valor que se trata junto al propio valor.

- Lista de valores: Se trata de un conjunto de valores similar a los vectores o listas implementados en los lenguajes de programación.

Como puede verse, cualquiera de estas estructuras resulta universal, haciendo posible que todos los lenguajes de programación lo soporten de diferentes formas.

A continuación, podemos ver un ejemplo de una colección de datos representadas en formato JSON:

```
{
  "else": "yes",
  "condition": "$cond3 and $cond4",
  "name": "IfCondB",
  "type": "IF"
}
```

Este pequeño fragmento JSON se corresponde con una actividad *If* del lenguaje WS-BPEL. En este caso, se nos indicaría que la actividad es de tipo If, su nombre es IfCondB, en la condición intervendrían dos variables y contendría un bloque del tipo *else*.

1.7. Glosario

A continuación se definen una serie de acrónimos y definiciones que resultan útiles a la hora de entender ciertos apartados de este TFG.

1.7.1. Acrónimos

TFG Trabajo de Fin de Grado

PM Prueba Metamórfica

RM Relaciones Metamórficas

SW Servicios Web

CSV Comma Separated Values

JSON JavaScript Object Notation

WS-BPEL Web Services Business Process Execution Language

WSDL Web Services Description Language

BPTS BPELUnit Test Suite

1.7.2. Definiciones

- **Composición WS-BPEL:** Conjunto de servicios que forman un Servicio Web a partir de otros ya existentes. Este tipo de composiciones utiliza módulos en lenguaje WS-BPEL.
- **Servicio Web:** Utilidad que permite el intercambio de datos entre sistemas mediante un conjunto específico de protocolos y estándares.
- **XML:** Meta-lenguaje (no marcado) que permite definir lenguajes de marcado específicos para un uso determinado.
- **SOAP:** Protocolo estandarizado que define la comunicación entre objetos de diferentes procesos mediante el intercambio de datos XML.

2. Técnica de prueba metamórfica

En este capítulo se define la técnica de la prueba metamórfica de manera teórica, así como, su aplicación a composiciones WS-BPEL 2.0.

Esta técnica se basa en el concepto de *relaciones metamórficas*. Se pretende aplicar esta técnica a composiciones WS-BPEL y probar su eficacia a través de la prueba de mutaciones.

2.1. Definición

La técnica de la prueba metamórfica [8] es una técnica de prueba de software que trata de detectar los errores que se cometen durante la etapa de diseño y desarrollo del software, y cuyo objetivo es aliviar el problema del oráculo.

El problema del oráculo es uno de los grandes retos a los que las técnicas de prueba de software se enfrenta hoy en día. Se considera oráculo aquella entidad capaz determinar la corrección de unos determinados datos de salida tras la ejecución de un software respecto a unos datos de entrada. Esta entidad puede ser desde un procedimiento o un programa de prueba, hasta una persona o equipo de personas expertas capaz de determinar la validez de estos datos.

Este problema no se da en el conjunto de programas probables -*testable programs*-, que son aquellos que disponen de un modo para probar su corrección, sino que se da en el conjunto de programas no probables -*non-testable programs*-, y consiste en la falta de oráculo, ya sea porque no existe, o bien, en el caso de existir, resulta prácticamente imposible comprobar la corrección del conjunto de datos de salida. [25]

La aplicación de la técnica de prueba metamórfica ha sido ampliamente alabada a la hora de aliviar este problema, ya que es un método diseñado para probar aquel software en los que el oráculo no puede aplicarse.

Así mismo, la prueba metamórfica se utiliza para comprobar la calidad de un conjunto de casos de prueba. Tras su aplicación inicial a un conjunto determinado de casos de prueba sobre programas que contienen errores o modificaciones (mutantes), podemos obtener casos *de éxito* y casos *erróneos*. [9] Los casos de prueba denominados erróneos son aquellos que han detectado errores específicos en el programa bajo prueba (utilizando alguna técnica de prueba de software), así mismo, los denomina-

dos de éxito, son aquellos que no han detectado ningún error en el programa tras su ejecución sobre ellos. Los casos de prueba de éxito son objeto de estudio, puesto que al aplicar la técnica de la prueba metamórfica sobre ellos podemos detectar errores que anteriormente no habían sido encontrados.

Por lo tanto, podemos ver que se puede aplicar de nuevo la técnica de prueba metamórfica sobre los resultados obtenidos tras la aplicación de la misma.

Existe relación entre cada *caso de prueba inicial* y cada caso generado tras la aplicación de la técnica (*caso de prueba siguiente*). Esta relación forma parte de la técnica, y se denomina *relación metamórfica*, representando una propiedad que es necesaria. Si un caso de prueba inicial no detecta un error existente y tras la aplicación de la técnica, su caso de prueba siguiente sí detecta dicho error, esa relación metamórfica ha servido para detectarlo.

2.2. Relaciones Metamórficas

El concepto de prueba metamórfica está muy relacionado con las relaciones metamórficas. Podríamos definirlas de la siguiente manera: *propiedades existentes sobre el conjunto de entradas y sus correspondientes resultados para múltiples evaluaciones de una función* [3]. Es decir, partiendo de un conjunto particular de entradas correctas, se espera un conjunto de salidas que cumplan unas propiedades correspondientes que serán evaluadas. Mediante la aplicación de las relaciones metamórficas se busca obtener nuevos casos de prueba a partir de un conjunto inicial de partida. El objetivo es que estos nuevos casos de prueba detecten los errores que los casos iniciales de los que partían no detectaban.

Veamos esta aplicación directamente con un ejemplo. Consideremos la composición WS-BPEL **Triángulo** tratada en la Sección 1.5, cuya ejecución permite determinar la geometría de un triángulo existente dados sus lados.

Pensemos en un triángulo de entrada, definido por una terna que representa los valores de los lados que lo componen, $T_1 = (4, 4, 4)$. Aplicando esta entrada a la composición Triángulo, obtendríamos como resultado 3, lo que nos indica que estamos ante un triángulo *equilátero*.

Es sabido que la geometría de un triángulo depende directamente de la proporcionalidad de sus lados, es decir, el tamaño de los mismos no influirá a menos que difieran entre sí. Por lo tanto, si multiplicamos todos los lados del triángulo de entrada por el mismo numero natural, como el 2 por ejemplo, obtendremos un segundo triángulo cuyos lados serían representados por $T_1' = (8, 8, 8)$. Este nuevo triángulo sigue manteniendo su proporción, siendo todos sus lados del mismo tamaño, por lo

que sabemos que si aplicamos la composición Triángulo a este nuevo triángulo, el resultado volverá a ser equilátero. Por lo tanto, puede verse que T_1 y T_1' están relacionados entre sí, y que al multiplicar por un número natural todos los lados de T_1 el resultado de la aplicación de la composición no variará. De esta manera, obtenemos una relación metamórfica, que formalmente estará representada de la siguiente forma:

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

Aplicando esta relación metamórfica obtenemos un caso de prueba nuevo, con el que comprobaremos que si al ejecutarlo no se cumple la propiedad mencionada, obteniendo un resultado distinto al esperado, estaremos detectando un error en el programa a probar, puesto que la propiedad es necesaria.

2.3. Técnica de prueba de mutaciones

La técnica de prueba de mutaciones [10] permite evaluar la calidad de un conjunto de casos de prueba. Mediante la realización de ligeros cambios en el código del software que se quiere probar, llamados *mutaciones*, y la aplicación de un conjunto de *operadores de mutación* específicos para cada lenguaje de programación, obtendremos un conjunto de programas similar al programa que se quiere probar, pero con pequeñas variaciones insertadas, las cuales deberán ser detectados por los casos de prueba al ejecutarse sobre los mutantes.
Un buen conjunto de casos de prueba será capaz de distinguir el programa original de sus mutantes, aunque podrían darse casos en los que el mutante es imposible de detectar: mutantes equivalentes.

La técnica de prueba de mutaciones se utiliza como técnica auxiliar en las composiciones WS-BPEL para comprobar que se detectan errores de forma correcta durante la aplicación de la técnica metamórfica.

2.4. Aplicación a una composición

Para mostrar el funcionamiento de la aplicación de las RM, ilustraremos su comportamiento aplicando algunas de ellas a la composición *Triángulo* 1.5.
Toda composición WS-BPEL puede probarse con un conjunto de casos de prueba contenidos en un fichero BPTS. Los casos de prueba que se tendrán en cuenta para esta composición tendrán la siguiente forma:

$$(a, b, c, result)$$

Siendo *a, b* y *c* los lados del triángulo proporcionado, mientras que *result* se corresponde con el resultado esperado.

Estas son algunas de las RM obtenidas manualmente tras el análisis de la composición:

- **RM1:**

$$a_2 = a_1 \wedge b_2 = b_1 \wedge c_2 = 2 \cdot c_1 \wedge result_1 = 3 \Rightarrow result_2 = -2$$

- **RM2:**

$$a_2 = a_1 \wedge b_2 = 2 \cdot b_1 \wedge c_2 = 3 \cdot c_1 \wedge result_1 = 3 \Rightarrow result_2 = 1$$

- **RM3:**

$$a_2 = a_1 \wedge b_2 = b_1 \wedge result_1 = 2 \wedge c_2 \geq a_1 + b_1 \Rightarrow result_2 = -2$$

Puede observarse como en todas las relaciones obtenidas el formato se compone de un antecedente y un consecuente, compuestas por expresiones relacionales unidas con operadores lógicos. Algunas de estas expresiones contienen a su vez subexpresiones aritméticas.

Supongamos la existencia de un conjunto de casos de prueba para esta composición que toma los siguientes valores:

1. $(4, 4, 4, 3)$
2. $(4, 2, 4, 2)$
3. $(4, 3, 2, 1)$
4. $(4, 4, 8, -2)$

Aplicando la RM1 obtendríamos un conjunto de casos de prueba siguiente, que tendrían la siguiente forma:

1. $(4, 4, 8, -2)$
2. $(4, 2, 8, -2)$

3. $(4, 3, 4, -2)$
4. $(4, 4, 16, -2)$

De igual manera, podemos aplicar el resto de RM obteniendo un conjunto de casos de prueba siguiente similar. Aplicando la técnica de prueba metamórfica, se busca comprobar la validez de los casos de prueba siguientes. Si alguno de estos casos detectase un error en uno de los mutantes, diríamos que dicho mutante está muerto (ese caso de prueba lo ha matado), por lo que, en el caso de ser una composición real se corregiría el error detectado. Si no se detectase ningún error, el mutante seguiría vivo y podemos seguir aplicando la técnica de prueba metamórfica, generando nuevos casos de prueba siguientes, hasta que detectemos errores que no se habían detectado en los casos dep rueba iniciales, o bien, hasta que se cumpla un determinado criterio establecido.

3. Desarrollo del calendario

A lo largo del desarrollo de este proyecto se ha seguido la metodología iterativa basada en prototipos. Mediante reuniones periódicas, se han ido revisando prototipos funcionales donde se han ido implementando funcionalidades hasta conseguir una herramienta completa, así como correcta, ya que no se han desarrollado nuevas funcionalidades hasta que las ya implementadas funcionaban perfectamente.

El motivo de seguir esta metodología ha sido la naturaleza del proyecto, ya que se trata de un proyecto de investigación con el grupo de investigación UCASE.

3.1. Fases

A continuación se detallan todas las fases por las que ha pasado este proyecto.

3.1.1. Aprendizaje de tecnologías

Durante un tiempo se realizó una formación sobre tecnologías relacionadas, concretamente sobre la prueba de software y WS-BPEL con el objetivo de familiarizarme con las composiciones y herramientas disponibles.

Esta formación se ha realizado mediante el estudio de diferentes artículos publicados sobre el tema, reuniones periódicas con la tutora del proyecto y seminarios en el laboratorio UCASE.

3.1.2. Estudio de análisis de composiciones

Durante este periodo se estudiaron diferentes técnicas de análisis de composiciones, tanto por artículos científicos como estudiando proyectos anteriores.
Se mantuvieron reuniones con personas del grupo UCASE y alumnos egresados con el objetivo de obtener la mayor información posible para poder analizar las composiciones de la manera mas eficaz y eficiente posible.

3.1.3. Herramienta

Tras el periodo de estudio, se planteó la posibilidad del desarrollo de una herramienta capaz de ayudar al proceso de obtención de RM. Se decidió que lo ideal sería desarrollar una herramienta que mediante el análisis apropiado de los elementos necesarios se pudiese obtener información de la composición que ayude a otras herramientas o a la persona encargada de esta etapa a obtener las relaciones metamórficas de una manera más eficiente y eficaz.

3.1.4. Implementación del análisis

Tras recabar toda la información posible, se creó el proyecto utilizando Eclipse. Mediante el lenguaje de programación Java buscamos la manera de explorar el contenido de los casos de prueba para así poder filtrar lo que necesitásemos. La herramienta test-generator-autoseed [15] implementa unos módulos específicos que permiten abrir ficheros BPEL y explorar sus elementos mediante un conjunto de *getters* y *setters*, por lo que se decidió que un buen método para realizar esta tarea sería heredar este módulo y darle el uso específico que estamos buscando.

Al finalizar esta etapa ya existía el **primer prototipo**, capaz de abrir un fichero BPEL y mostrar información detallada de los elementos que contiene.

3.1.5. Informe

Durante este periodo se estudiaron las distintas posibilidades ofrecidas por el lenguaje para que la información obtenida de las composiciones WS-BPEL fueran escritas en un informe.
En este punto se decidió que lo mejor sería generar dos informes, uno que pudiese ser automatizado por otras herramientas y otro más legible que pudiese ayudar a la persona encargada del análisis manual de las composiciones.

En este punto se decidió que el informe para el proceso manual podría producirse en texto plano, escrito con un lenguaje adecuado, sin embargo, el formato para la automatización fue objeto de debate al existir multitud de formatos y lenguajes que podían satisfacer las necesidades. Aunque se plantearon otros formatos como plantillas Velocity, finalmente decidimos utilizar un conjunto de ficheros CSV.

Al finalizar esta etapa ya existía el **segundo prototipo**, cuya novedad era la generación de un pequeño informe en texto plano, así como una serie de ficheros CSV con información básica.

3.1.6. Análisis de los casos de prueba

Tras ser capaces de analizar las composiciones WS-BPEL y generar una serie de informes sencillos, se empezó con el desarrollo de un módulo de análisis para los casos de prueba BPTS.
En primer lugar, pudimos ver que las librerías utilizadas para las composiciones WS-BPEL, que eran perfectamente válidas y eficientes para el tratamiento de las actividades, no lo eran sin embargo, para el tratamiento de los casos de prueba BPTS. Por lo que debía hacerse con otras herramientas.
Tras un periodo de estudio de posibilidades, se decidió utilizar las herramientas estándar que provee el lenguaje Java, pues con el tratamiento adecuado del fichero podemos utilizar una serie de librerías que transforman los elementos en formato XML en objetos propios del lenguaje que podemos utilizar para explorar la información y generar el informe deseado en el formato que necesitemos.

Al finalizar esta etapa ya existía el **tercer prototipo**, cuya novedad era la generación de otro informe en texto plano con la información contenida en los casos de prueba.

3.1.7. Artículo para JCIS 2016

Las Jornadas de Ciencia e Ingeniería de Servicios (JCIS) son un foro de discusión e intercambio de conocimiento y experiencias. Estas jornadas están centradas en los nuevos avances científicos y en las tecnologías existentes en torno a la computación orientada a servicios y procesos de negocio, así como las nuevas prácticas de ingeniería de servicios y las lecciones aprendidas mediante experiencias reales.
JCIS propicia el encuentro de las comunidades relacionadas con el ámbito de la Ingeniería de Servicios, concretamente en Servicios Web, SOA y Procesos de Negocio.

Como podemos ver, este trabajo encaja en la temática de estas jornadas, por lo que se decidió escribir y presentar un artículo que trata sobre la idea principal de este trabajo, así como algunos ejemplos de informes generados por la herramienta y como se está llevando a cabo todo el proceso.
Durante esta fase, se determinó que la mejor opción para generar un informe que automatice el proceso de obtención de relaciones metamórficas era utilizar ficheros en formato JSON [13], siendo estos los más adaptables a nuestros requisitos con la mayor compatibilidad posible.

El artículo ha sido aceptado como artículo corto y será presentado en Salamanca durante la celebración de las jornadas en septiembre de 2016.

3.1.8. Informes JSON

Durante este periodo se desarrolló el módulo de la herramienta correspondiente al informe que facilitará la automaticación de obtención de RM.
Se determinó que la mejor opción es utilizar un *array* de JSON que contenga toda la información recogida en el análisis y sea presentado en un fichero *.json*.

Al finalizar esta etapa ya existía el **cuarto prototipo**, cuya novedad era la generación de informes paralelos a los ya existentes con la información en JSON. Se desecharon los ficheros CSV y se hicieron distintas pruebas donde se cargaba en memoria la información escrita en los informes JSON.

3.1.9. Cruce de informes

Durante este periodo se estudió una posible discriminación de los elementos obtenidos, con el objetivo de proporcionar la información que más se ajuste a la tarea, siendo esta la más concreta que muestre directamente aquellos elementos más propensos a formar parte de las relaciones metamórficas.
Finalmente se decidió que la mejor opción era buscar aquella información que estuviese contenida tanto en las composiciones WS-BPEL como en los casos de prueba BPTS, ya que esta es más propensa a generar casos de prueba siguientes.

Al finalizar esta etapa ya existía el **quinto prototipo** cuya novedad era la generación de un tercer informe con toda la información obtenida del cruce de informes anteriores.

3.1.10. SGSOACS 2016

SGSOACS (Spanish-German Symposium on Applied Computer Science) es un evento organizado por el grupo de investigación UCASE de la Universidad de Cádiz donde se celebra la colaboración entre la Universidad de Cádiz y la Universidad de Ciencias Aplicadas de Frankfurt, en Alemania.
El objetivo de este simposio es reunir a investigadores y alumnos para que puedan exponer sus trabajos a todos junto con un turno de preguntas abiertas y debate donde todos pueden participar.

Los fundamentos de este trabajo y un estado actual fueron presentados en la tercera edición de este simposio en inglés el día 28 de junio de 2016, obteniendo un certificado y participando en el debate con el resto de asistentes, donde se recibieron buenos consejos y se obtuvieron ideas que han sido aplicadas a este proyecto.

Durante la asistencia a las otras ponencias se pudieron ver ciertas formas de pro-

gramación y algunas librerías del lenguaje Java que siendo totalmente compatibles con la herramienta desarrollada permiten que este sea mucho más eficiente. Se tomó nota de todos los cambios y se aplicó la mejora a las funciones ya existentes, reduciendo el tiempo de ejecución a más de la mitad y dando lugar al **sexto prototipo**. Además, en este evento se expuso un trabajo sobre EVOSuite [12], pudiéndose ver una demostración de la mano de su creador, respondiendo él mismo a todas las cuestiones planteadas. El conocimiento adquirido en este punto me animó a utilizar esta herramienta para generar las pruebas unitarias de la herramienta.

3.1.11. Mejora del código fuente

En esta etapa se aplicó la herramienta *Sonar*, encargada de evaluar el código Java de una aplicación dada, con el objetivo de mejorar el código fuente ya implementado. Tras varios análisis se ha conseguido un código de gran calidad que pasa todas las etapas de análisis de esta herramienta.

3.1.12. Documentación

A lo largo del desarrollo del proyecto se ha ido documentando todo el trabajo finalizado. Esta etapa es la última puesto que aunque ha sido desarrollada en conjunto con el proyecto, ha sido la última en finalizar.

3.2. Diagrama de Gantt

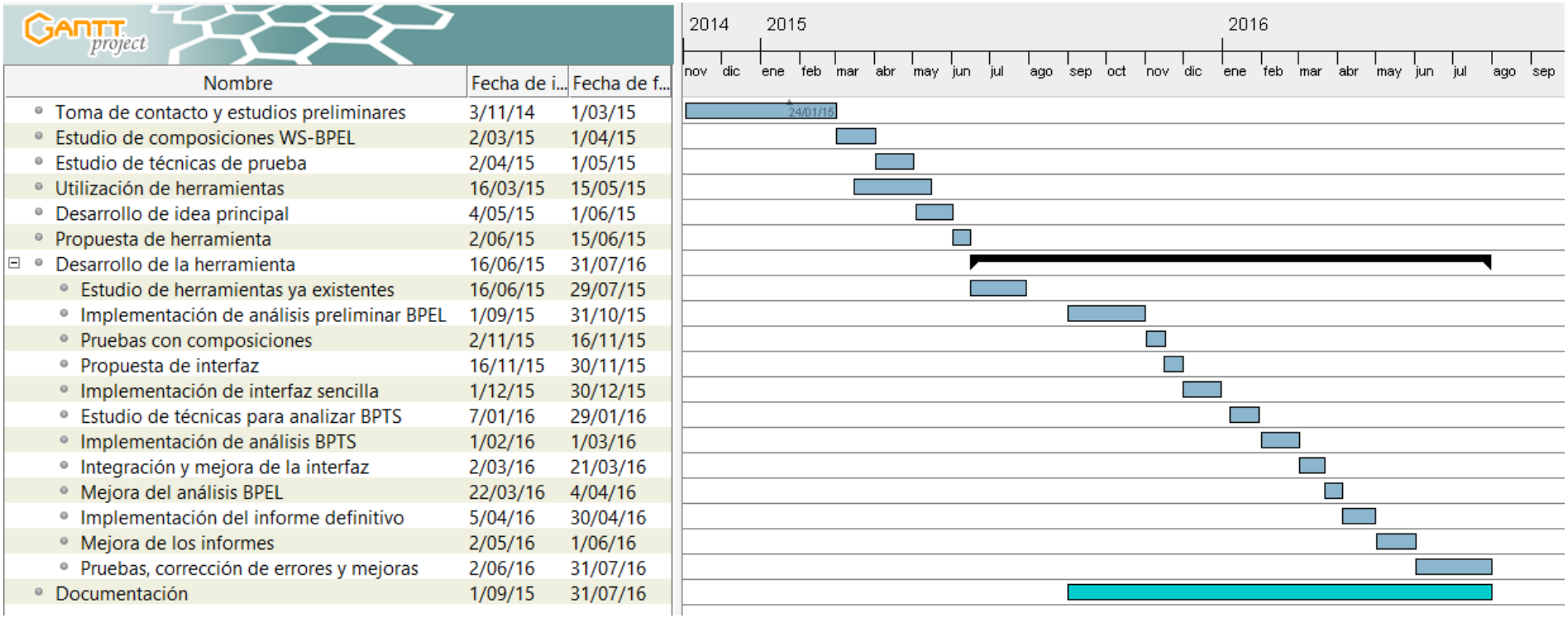

Figura 3.1.: Diagrama de Gantt del proyecto

4. Descripción general del proyecto

4.1. Perspectiva del producto

En este apartado se tratan los detalles fundamentales de la herramienta desarrollada en este trabajo, su interfaz y las características de sus funciones.

4.1.1. Entorno del producto

La herramienta **Analyzer4BPEL** es una herramienta orientada a la obtención y formateado de información obtenida de las composiciones WS-BPEL y sus casos de prueba para facilitar la implementación de relaciones metamórficas. A partir de la ubicación de la composición WS-BPEL, esta herramienta es capaz de detectar sus casos de prueba y generar información que tras un adecuado procesamiento, podrán ser utilizadas para la obtención automática de relaciones metamórficas.
La herramienta consta de los siguientes módulos:

- Módulo de análisis: una vez indicada la localización de la composición a analizar, se comienza un análisis en preorden de la composición y sus casos de prueba. Los resultados se envían de manera individual (por una parte los de la composición y por otra la de sus casos de prueba) al módulo de escritura.

- Módulo de unión: a partir del resultado del análisis, se hace una unión de la información en busca de elementos útiles que puedan llevar a la obtención de Relaciones Metamórficas.

- Módulo de escritura:

    1. Por una parte, se reciben los datos del analizador y se guarda la misma en dos ficheros distintos. El primero de ellos tendrá formato **JSON** para su posterior tratamiento de forma automática por otras herramientas. El segundo de ellos, estará escrito en texto plano utilizando el **lenguaje natural**, lo cual facilitará al usuario analista la tarea de análisis, dando información relevante acerca de la composición de forma más agradable.
    2. Por otra parte, se reciben los datos del módulo de unión y se genera un informe -igualmente en los dos formatos anteriores- con información sobre los elementos clave a la hora de generar Relaciones Metamórficas.

La herramienta **Analyzer4BPEL** toma como dependencias algunos módulos desarrollados para la aplicación **TestGenerator-Autoseed** [15]. Así mismo, la herramienta presenta algunas dependencias externas como Java 7.

4.1.2. Interfaz de usuario

Esta herramienta presenta una interfaz de usuario sencilla, con el objetivo de facilitar lo máximo posible la parte correspondiente de la fase de análisis y obtención de propiedades de las composiciones WS-BPEL. Así mismo, la interfaz ha sido diseñada a medida para las composiciones estudiadas de ejemplo, sin embargo, la herramienta está preparada para trabajar con otras composiciones WS-BPEL.

Puede observarse esta interfaz en la Figura 4.1

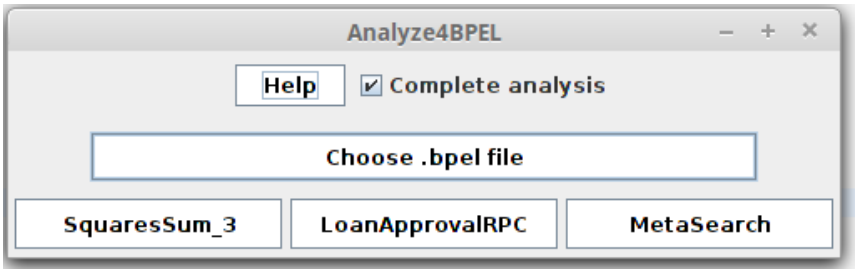

Figura 4.1.: Interfaz de Analyzer4BPEL

En el Anexo D pueden encontrarse detalles acerca del funcionamiento de cada elemento de la misma.

4.2. Funciones

Analyzer4BPEL incluye funciones relacionadas con la fase de análisis y obtención de propiedades de composiciones WS-BPEL. A continuación, se detallan todas ellas:

- Generar un conjunto de ficheros para una composición determinada, que ofrecen información sobre que elementos de la composición WS-BPEL aparecen en la misma, independientemente de su relevancia a la hora de obtener relaciones metamórficas.
- Generar un informe para una composición determinada, que ofrece información relevante sobre qué elementos de la composición WS-BPEL resultan importantes para la obtención de relaciones metamórficas, teniendo en cuenta el contenido y la interacción con sus casos de prueba.

4.3. Características del usuario

Esta herramienta sirve como ayuda para la fase de análisis y obtención de propiedades de una composición WS-BPEL para su aplicación en la técnica de pruebas metamórficas. Es por esto por lo que el usuario debe tener conocimiento de estos conceptos antes de empezar a trabajar con ella.

Un usuario que requiera utilizar esta herramienta deberá tener al menos conocimientos básicos de lo siguiente:

- Lenguaje WS-BPEL 2.0 [16]
- Técnica de Pruebas Metamórficas [8]
- Lenguaje JSON [13]

4.4. Restricciones generales

En este apartado se detallan las restricciones mínimas que tiene la herramienta desarrollada, incluyendo así las restricciones relacionadas con el conocimiento del usuario y las restricciones relacionadas con el hardware donde se ejecuta esta.

4.4.1. Control de versiones

Durante el desarrollo de este TFG se ha utilizado un sistema de control de versiones con el objetivo de llevar un control de los cambios del mismo, ya que cada iteración resultaba ser un prototipo que iba obteniendo funciones de manera incremental. De igual manera, ha servido como fuente de respaldo para poder revertir los cambios perjudiciales si fuese necesario.
El uso del control de versiones proporciona muchas ventajas, ya que permite que se compartan todos los componentes del proyecto, facilitando la revisión o edición por parte de otras personas sin necesidad de enviar los archivos anexados manualmente.

Para el trabajo con los artículos y las diferentes herramientas estudiadas hemos utilizado el sistema de control de versiones **svn** [18], además, hemos utilizado el sistema de control de versiones **git** [24], pudiendo trabajar así con los dos sistemas más populares y usados en el repositorio del grupo UCASE [20].

Pueden verse los comandos necesarios para el manejo de este TFG en el Anexo D.

4.4.2. Lenguajes de programación y tecnologías

Para el desarrollo de este TFG se han utilizado diversos lenguajes de programación, no solo para el desarrollo de la aplicación principal, sino para el estudio relacionado con el análisis y las tecnologías consideradas para el formato de salida de los informes producidos.
Estos lenguajes son los siguientes:

- Java: Lenguaje empleado para la implementación del código de la herramienta.
- WS-BPEL: Se analiza el código de las composiciones en este lenguaje.
- BPELUnit: Se analiza el código de los ficheros BPTS.
- XPath: Utilizado para el recorrido de algunos elementos en los BPTS.
- JSON: Utilizado para un formato de informe.

4.4.3. Herramientas

Para el desarrollo de este TFG se han utilizado una serie de herramientas que han servido de manera directa para alcanzar el objetivo propuesto. Algunas de ellas han sido utilizadas de manera auxiliar para comprobar la validez del proceso implementado.
Las herramientas utilizadas son las siguientes:

- Eclipse [2]: plataforma de desarrollo de software de código abierto multiplataforma, ampliamente utilizada para el desarrollo de aplicaciones en distintos lenguajes, aunque principalmente utilizado para Java.
  Además, esta herramienta ha sido utilizada con una serie de plugins específicos para WS-BPEL [1] con el objetivo de estudiar la ejecución de una serie de composiciones y desarrollar algunos elementos propios.
- test-generator-autoseed [15]: herramienta utilizada para la generación automática de casos de prueba mediante siembra automática en composiciones WS-BPEL. Entre otras funciones, esta herramienta provee de un conjunto de funciones *getters y setters* capaz de manejar de forma rápida y eficiente los datos relacionados con las actividades en WS-BPEL. Estas funciones han sido heredadas para su implementación en el proceso de obtención de información de las composiciones WS-BPEL.
- XMLEye [11]: Visor de documentos XML con interfaz gráfica, utilizado para una mejor comprensión de los elementos contenidos en las composiciones y sus casos de prueba mediante una representación amigable.

4.4.4. Sistemas operativos y hardware

La herramienta presentada en este TFG ha sido desarrollada y probada en su totalidad en GNU-Linux. Concretamente, se ha utilizado Linux Mint 17 en su versión basada en Ubuntu.

Para su ejecución, los requisitos son muy bajos, ya que solo es necesario trabajar sobre los ficheros de las composiciones sin ejecutarlos. Se recomienda disponer de un equipo con un mínimo de 512 MiB de memoria RAM para que la ejecución no se demore más tiempo del esperado.

5. Desarrollo del proyecto

En este capítulo se muestra todo el proceso seguido hasta alcanzar el estado final de este TFG. Concretamente puede verse desarrollo del proyecto desde los requisitos iniciales, pasando por el análisis y el diseño del sistema, así como, ciertos detalles de la implementación y como se ha probado y validado el producto final.

5.1. Modelo de ciclo de vida

Durante el desarrollo de este TFG se ha seguido un modelo de ciclo de vida iterativo e incremental.

El modelo de ciclo de vida iterativo e incremental consiste en un conjunto de tareas que deben realizarse agrupadas en etapas que se repiten continuamente. Dichas etapas son las llamadas iteraciones. En cada iteración el software irá mejorando, puesto que las etapas incluyen las fases de análisis, diseño y pruebas, por lo que estas se realizan continuamente a lo largo del desarrollo del software, a diferencia de otros modelos de ciclo de vida, donde solo ocurren una vez.
En cada iteración se parte de una serie de requisitos y objetivos que deben cumplirse, por lo que al final del tiempo establecido se obtendrá un prototipo funcional del software desarrollado, de esta manera se puede obtener un producto real al final de cada iteración, que irá mejorando con el paso de cada una de ellas hasta llegar al resultado final esperado.
Este modelo pretende solucionar algunos de los problemas surgidos en el modelo de ciclo de vida en cascada, donde las etapas del proceso de desarrollo del software se encuentran totalmente diferenciadas y no se continuará con el proceso siguiente si el anterior no ha sido totalmente satisfecho.

Este modelo de ciclo de vida se adapta a las necesidades de este TFG. Durante todo el tiempo se han ido desarrollando reuniones cada dos semanas, donde se han especificado una serie de requisitos y objetivos a cumplir de cara a la siguiente reunión. En cada reunión se ha presentado una versión de la herramienta, cuyas funciones han ido incrementando y mejorando en cada iteración, de manera que desde una fase temprana se ha podido utilizar una versión funcional sin tener que esperar a la finalización del desarrollo, permitiendo además una flexibilidad respecto

a la definición de los requisitos, los cuales han ido definiéndose según las necesidades que han ido surgiendo.

En la Figura 5.1 puede verse de manera gráfica como se desarrollan las distintas iteraciones.

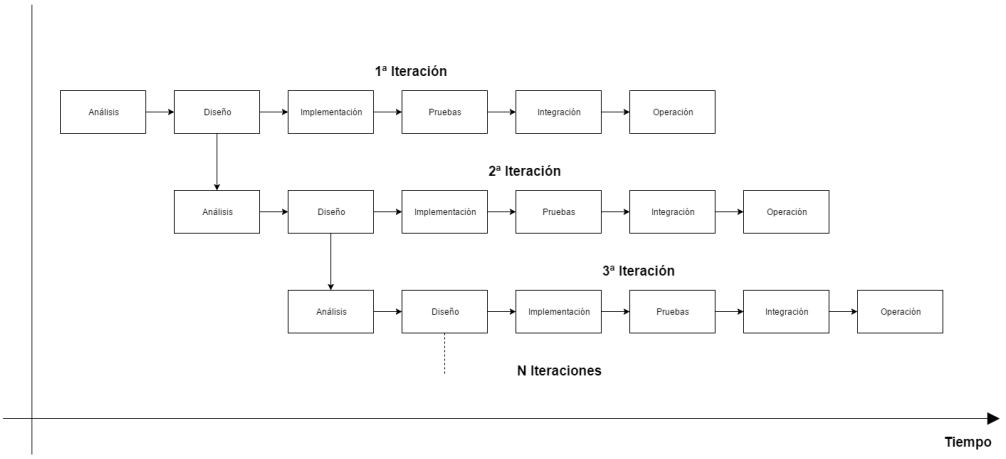

Figura 5.1.: Modelo de ciclo de vida iterativo e incremental

5.2. Requisitos

A continuación se detallan todos los requisitos del proyecto desarrollado.

5.2.1. Funcionales

- Analizar composiciones WS-BPEL: consiste en realizar un análisis del código de la composición WS-BPEL de manera que se busque y seleccione información que ayude a la obtención de RM para su posterior aplicación en la técnica de prueba metamórfica. Debe generarse un informe que recoja de manera detallada toda la información obtenida del análisis, descartando toda la información que no de pistas o no resulte de utilidad en la búsqueda de RM.

- Analizar casos de prueba BPTS: consiste en realizar un análisis del código de los casos de prueba BPTS que acompañan a las composiciones WS-BPEL. Este análisis busca y selecciona aquellos elementos utilizados en la búsqueda de RM, de manera que se desechan todos los elementos de control típicos del lenguaje, así como aquellos elementos que no intervienen en el desarrollo directo de la composición WS-BPEL. Debe generarse un informe que recoja

de manera detallada toda la información obtenida del análisis de una manera entendible y estructurada, que proporcione una ventaja respecto a la lectura directa del código XML.

- Buscar información coincidente entre la composición WS-BPEL y los casos de prueba BPTS: consiste en utilizar los análisis generados de la composición WS-BPEL y sus casos de prueba para buscar información coincidente en ambos, de manera que se obtendrá un informe con toda la información utilizada en la composición WS-BPEL que tiene presencia en los casos de prueba BPTS. Debe generarse un informe que recoja toda la información de la composición WS-BPEL que aparece en los casos de prueba BPTS, desechando todos los elementos anteriormente obtenidos que no aparecen en ambos elementos.

- Generar un informe en formato JSON con la información obtenida: debe obtenerse toda la información recogida de los análisis y presentarse como un fichero en formato JSON, donde los valores estarán organizados como *arrays* de JSON.

- Generar un informe en formato TXT con la información obtenida: debe obtenerse toda la información recogida de los análisis y presentarse como un fichero en formato TXT, donde los valores estarán formateados con el objetivo de facilitar la lectura manual de los resultados obtenidos.

5.2.2. De información

- El sistema debe guardar una ruta que se utilizará para dar salida a los informes, siendo por defecto el directorio *home* del usuario.

- De las composiciones WS-BPEL debe guardarse información de sus actividades, concretamente se guarda el *nombre* y el *tipo* de la actividad en todos los casos, mientras que dependiendo de la actividad se guardará la información particular de la misma.

- De los casos de prueba BPTS debe guardarse toda la información de los casos de prueba, concretamente se guarda el *nombre* de la etiqueta utilizada y su *valor*, divididos por los distintos casos de prueba que puede haber en un mismo fichero.

- Deben guardarse dos ficheros, en formato JSON y TXT donde se almacenará el resultado de las distintas etapas del análisis.

5.2.3. Reglas de negocio

La información obtenida de los distintos elementos debe contener siempre un valor y un nombre para el mismo, de manera que si existen valores sueltos (etiquetas null), se almacenarán para su estudio manual, pero no serán tenidos en cuenta en el informe final.

5.2.4. Interfaz

La herramienta **Analyze4BPEL** se utiliza como asistencia para la obtención de RM. Por lo tanto, los informes generados deben presentarse en formatos que permitan su automatización por otras herramientas a la vez que facilita el análisis manual de la información contenida en los distintos elementos.
Uno de los formatos de salida debe ser **JSON**, puesto que es uno de los más utilizados y compatibles con una gran diversidad de herramientas y lenguajes de programación, mientras que el otro de los formatos debe ser un formato TXT, donde los datos estarán formateados de manera que resulte fácil de leer para la persona correspondiente.

5.2.5. No funcionales

- Rendimiento: El análisis de las composiciones, sus casos de prueba y el cruce de información debe hacerse en el menor tiempo posible, de manera que el tiempo de espera para la obtención de los resultados no suponga un inconveniente, independientemente de la longitud o complejidad de la implementación de los distintos elementos analizados.

- Usabilidad: La utilización de la herramienta debe ser muy sencilla, permitiendo obtener resultados válidos con la mínima interacción del usuario.

- Mantenimiento: La herramienta debe ser fácil de mantener y modificar, con el objetivo de que su ampliación o corrección pueda ser realizada en el menor tiempo posible por cualquier persona implicada en el proyecto.

- Fiabilidad: Los resultados obtenidos en el informe deben ser realmente útiles para la obtención de RM y no contener ningún error en los valores indicados.

5.3. Análisis del sistema

En esta sección se ven los distintos casos de uso que se han tomado como referencia para diseñar la herramienta de análisis. Adicionalmente se muestran las diferentes

alternativas que el usuario podría encontrarse durante su uso, así como, el conjunto de diagramas de secuencia y el modelo del diseño utilizado.

5.3.1. Casos de uso

El modelo de casos de uso de la herramienta desarrollada en este TFG puede verse en la Figura 5.2.
A continuación se muestra de forma detallada la especificación de cada uno de los casos de uso.

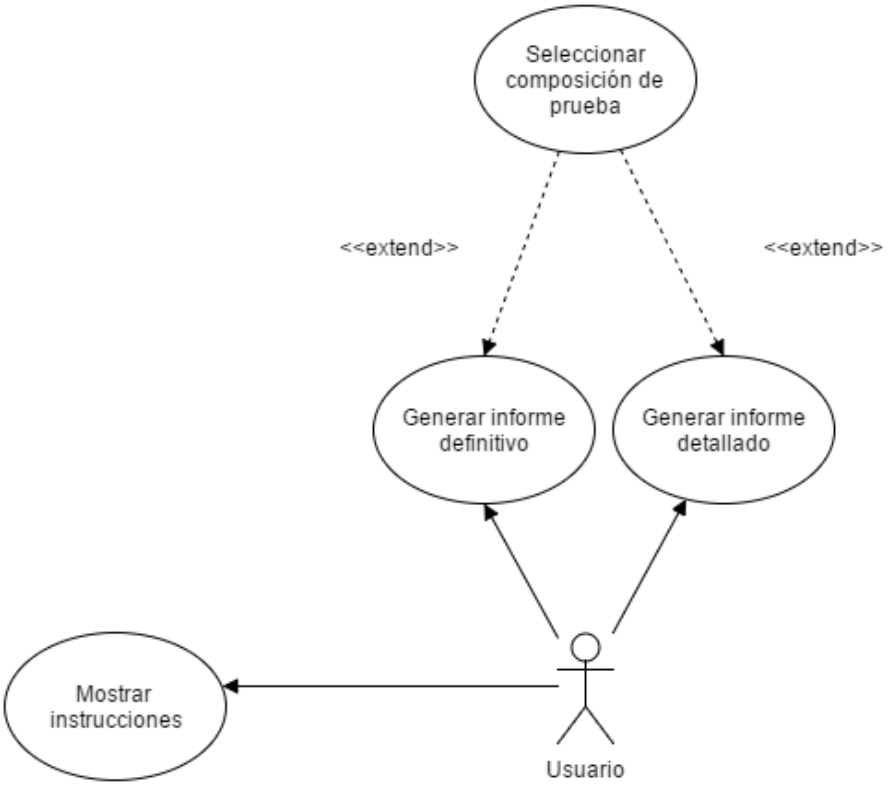

Figura 5.2.: Diagrama de casos de uso

5.3.1.1. Mostrar instrucciones

Actores: El usuario que utiliza la herramienta.

Precondiciones: El usuario no sabe utilizar la herramienta o quiere consultar las instrucciones de funcionamiento.

Postcondiciones: Se muestra una ventana con instrucciones de uso de la herramienta, así como de todos los parámetros y la ruta donde se generarán los informes correspondientes.

Escenario principal:

1. El usuario pulsa en la opción "Mostrar instrucciones de uso".
2. La herramienta muestra una nueva ventana con instrucciones de uso de si misma.

Escenario alternativo: No existe camino alternativo en este caso de uso.

Extensiones: No existen extensiones en este caso de uso.

5.3.1.2. Seleccionar composición de prueba

Actores: Este caso de uso es abstracto, por lo que no es utilizado por ningún actor. Puede ser activado opcionalmente en los casos de uso 5.3.1.3 y 5.3.1.4.

Precondiciones: El usuario quiere utilizar una de las composiciones propuestas para ver el funcionamiento del sistema.

Postcondiciones: Se genera un informe en formato JSON y TXT con información sobre los elementos de la composición WS-BPEL seleccionada y los elementos del caso de prueba BPTS correspondiente.

Escenario principal:

1. El usuario selecciona la opción 'LoanApproval'.
2. El sistema asigna como fichero .bpel el de la composición seleccionada y un fichero .bpts de prueba específico para esa composición.

Escenario alternativo:

1 a) El usuario selecciona la opción 'MetaSearch'.

1. Se continúa con el paso 2.

1 b) El usuario selecciona la opción 'SquaresSum'.

2. Se continúa con el paso 2.

Extensiones: No existen extensiones en este caso de uso.

5.3.1.3. Generar informe definitivo

Actores: El usuario que utiliza la herramienta.

Precondiciones: El usuario necesita obtener un informe completo sobre los elementos de la composición WS-BPEL que aparecen en el caso de prueba y resultan útiles en la obtención de relaciones metamórficas.

Postcondiciones: Se genera un informe en formato JSON y TXT con información sobre los elementos de la composición WS-BPEL que aparecen en el caso de prueba BPTS correspondiente y resultan útiles para la obtención de relaciones metamórficas.

Escenario principal:

1. El usuario pulsa en la opción "Select .bpel file".
2. El sistema muestra una nueva ventana con un explorador de archivos en el directorio */home/* del usuario.
3. El usuario navega por la interfaz, selecciona un fichero .bpel de su máquina y pulsa el botón 'Aceptar'.
4. El sistema muestra una nueva ventana con el explorador de archivos abierto en el mismo directorio donde se ha seleccionado el fichero .bpel.
5. El usuario navega por la interfaz, selecciona un fichero .bpts de su máquina y pulsa el botón 'Aceptar'.
6. El sistema genera el informe, muestra una ventana de éxito y abre el informe en el editor de textos predeterminado.

Escenario alternativo:

3 a) El usuario pulsa el botón 'Cancelar'.

1. Se cierra la ventana y el sistema vuelve a la ventana principal.

3 b) El usuario selecciona un fichero que no es .bpel

2. El sistema muestra un mensaje de error y vuelve a la ventana principal.

5 a) El usuario pulsa el botón 'Cancelar'.

3. Se cierra la ventana y el sistema vuelve a la ventana principal.

5 b) El usuario selecciona un fichero que no es .bpts

4. El sistema muestra un mensaje de error y vuelve a la ventana principal.

Extensiones:

1 a) El usuario selecciona la opción de utilizar una de las composiciones de prueba pulsando el botón con su nombre.

1. El sistema realiza el caso de uso 5.3.1.2.
2. El caso de uso continúa en el paso 6.

5.3.1.4. Generar informe detallado

Actores: El usuario que utiliza la herramienta.

Precondiciones: El usuario necesita obtener un informe completo sobre los elementos de la composición WS-BPEL y su caso de prueba sin discriminación de elementos y de manera separada.

Postcondiciones: Se genera un informe en formato JSON y TXT con información sobre los elementos de la composición WS-BPEL y los elementos del caso de prueba BPTS correspondiente.

Escenario principal:

1. El usuario selecciona la opción 'Informe preliminar'.
2. El usuario pulsa en la opción "Select .bpel file".
3. El sistema muestra una nueva ventana con un explorador de archivos en el directorio */home/* del usuario.
4. El usuario navega por la interfaz, selecciona un fichero .bpel de su máquina y pulsa el botón 'Aceptar'.
5. El sistema muestra una nueva ventana con el explorador de archivos abierto en el mismo directorio donde se ha seleccionado el fichero .bpel.
6. El usuario navega por la interfaz, selecciona un fichero .bpts de su máquina y pulsa el botón 'Aceptar'.
7. El sistema genera el informe, muestra una ventana de éxito y abre el informe en el editor de textos predeterminado.

Escenario alternativo:

1 a) El usuario no selecciona la opción.

1. Se ejecuta el caso de uso 5.3.1.3.

3 a) El usuario pulsa el botón 'Cancelar'.

2. Se cierra la ventana y el sistema vuelve a la ventana principal.

3 b) El usuario selecciona un fichero que no es .bpel

3. El sistema muestra un mensaje de error y vuelve a la ventana principal.

5 a) El usuario pulsa el botón 'Cancelar'.

4. Se cierra la ventana y el sistema vuelve a la ventana principal.

5 b) El usuario selecciona un fichero que no es .bpts

5. El sistema muestra un mensaje de error y vuelve a la ventana principal.

Extensiones:

1 a) El usuario selecciona la opción de utilizar una de las composiciones de prueba pulsando el botón con su nombre.

1. El sistema realiza el caso de uso 5.3.1.2.
2. El caso de uso continúa en el paso 6.

5.3.2. Diagramas de secuencia

A continuación se muestran los diagramas de secuencia correspondientes con cada caso de uso mostrado anteriormente en la sección 5.3.1.
La Figura 5.3 se corresponde con el caso de uso 5.3.1.1, la Figura 5.4 se corresponde con el caso de uso 5.3.1.2, la Figura 5.5 se corresponde con el caso de uso 5.3.1.3 y la Figura 5.6 se corresponde con el caso de uso 5.3.1.4.

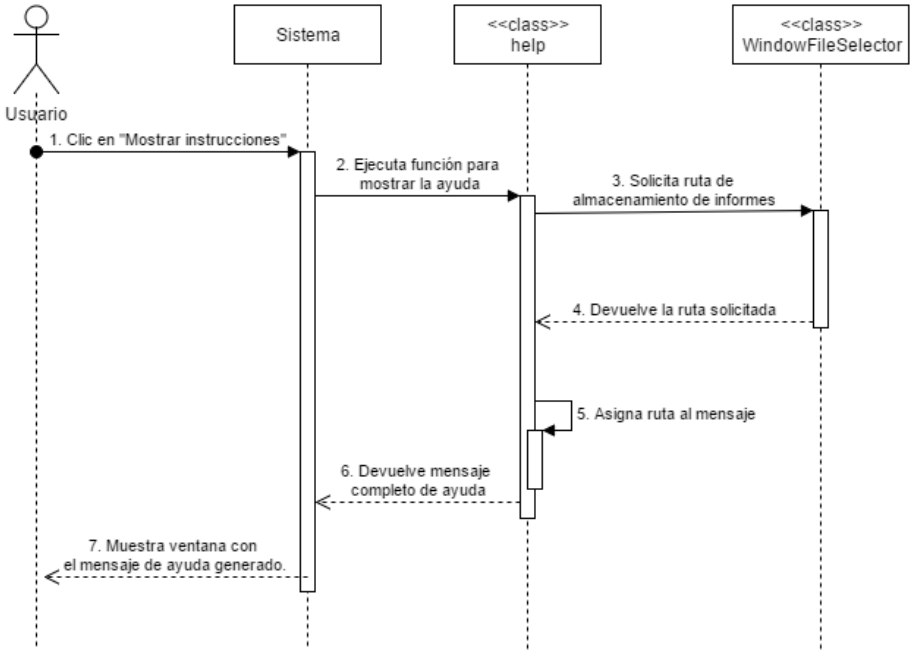

Figura 5.3.: Diagrama de secuencia de 'Mostrar instrucciones'

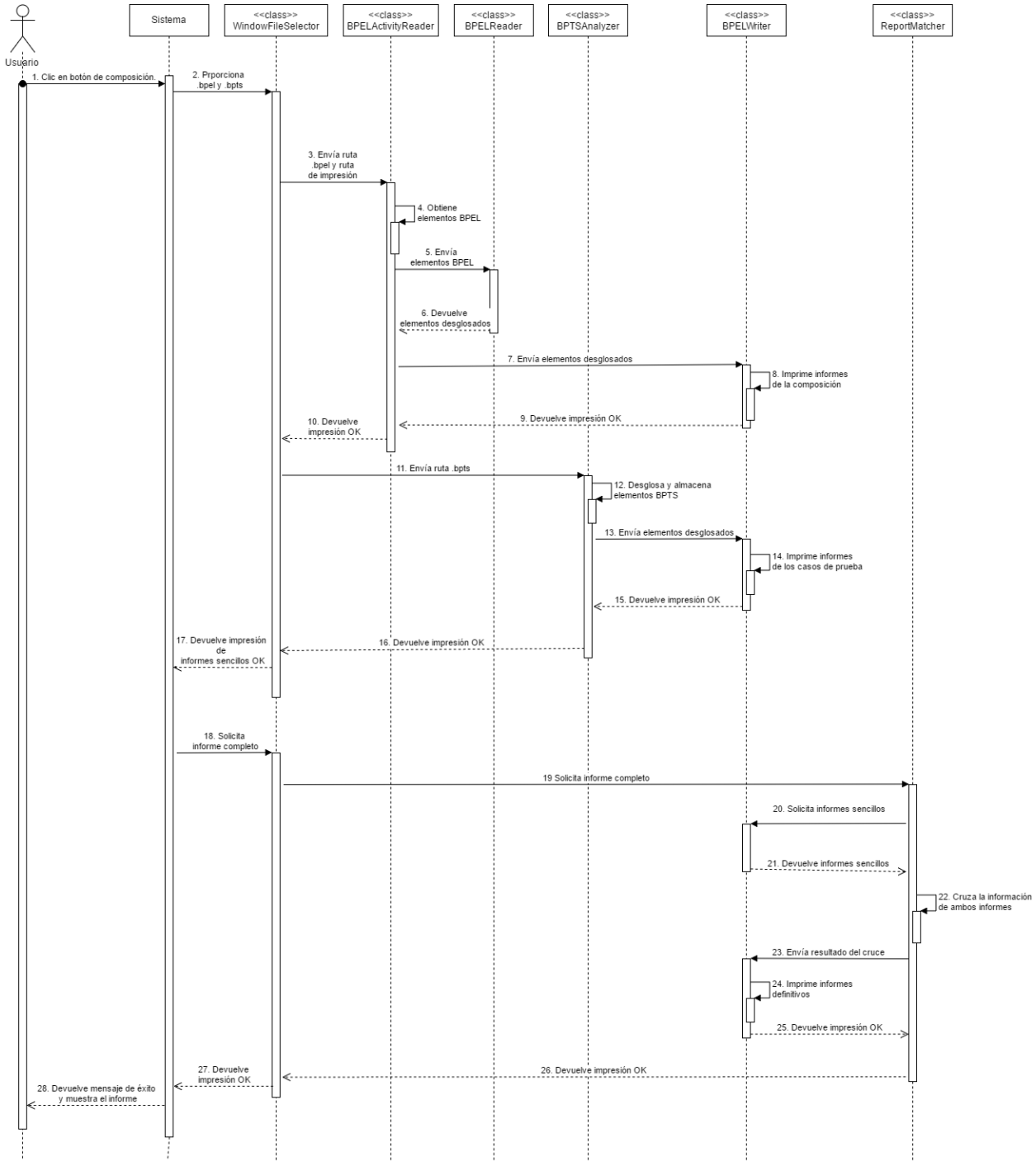

Figura 5.4.: Diagrama de secuencia de 'Seleccionar composición de prueba'

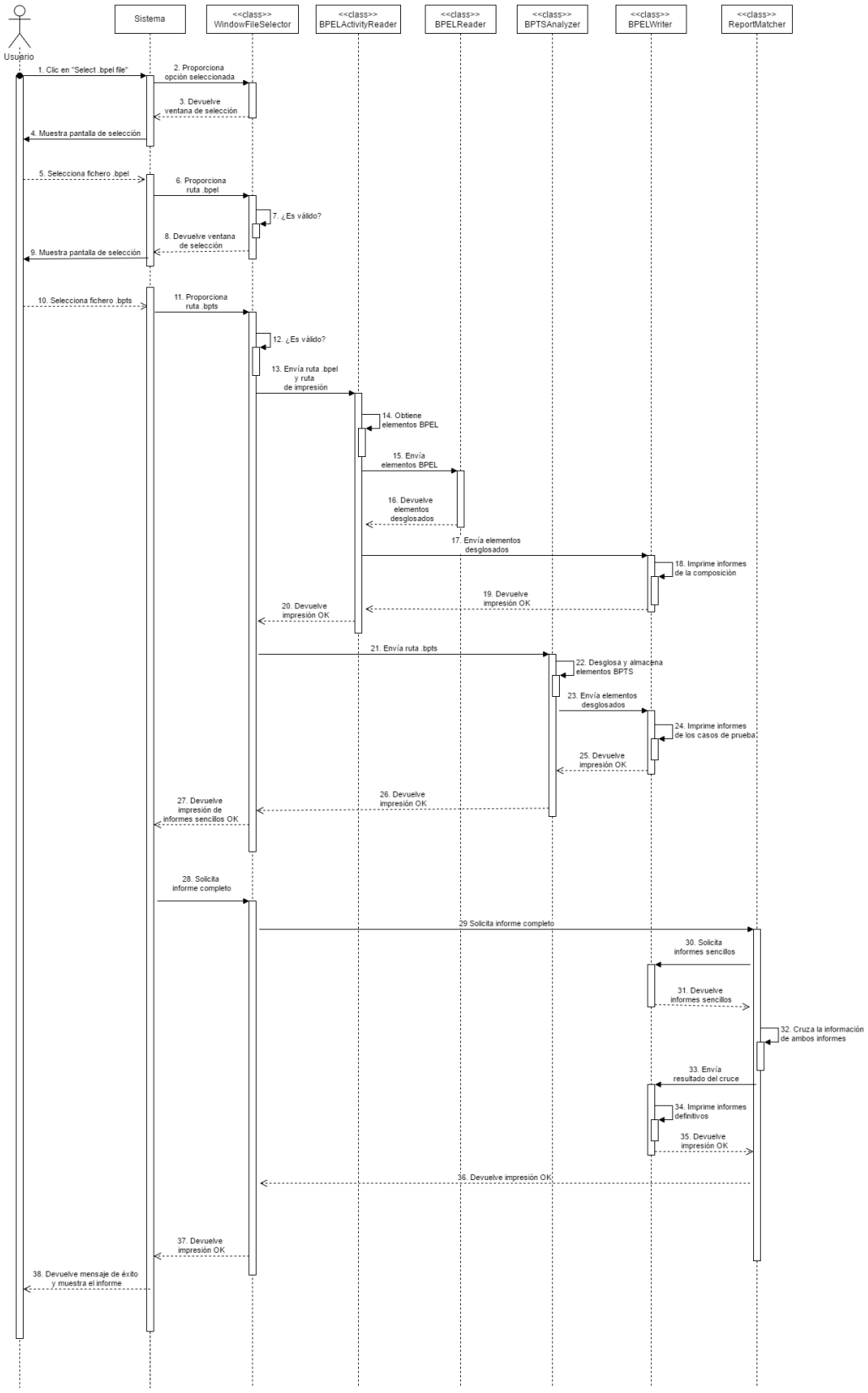

Figura 5.5.: Diagrama de secuencia de 'Generar informe definitivo'

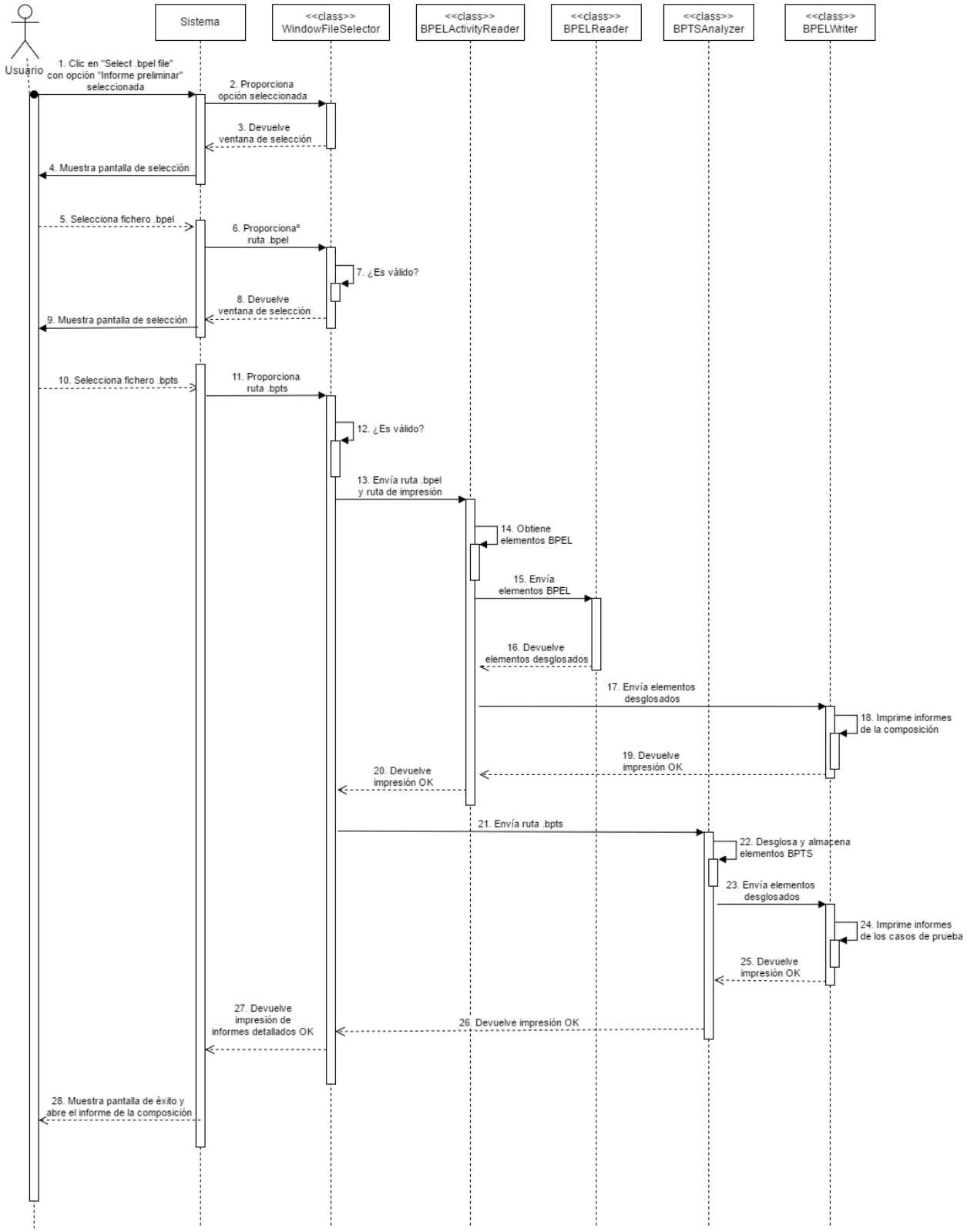

Figura 5.6.: Diagrama de secuencia de 'Generar informe detallado'

5.4. Diseño del sistema

En esta sección se detalla toda la estructura de la herramienta desarrollada en este TFG. Puede verse por completo la estructura interna de la aplicación y el esquema seguido para el diseño, así como, una justificación de todas las decisiones tomadas.

5.4.1. Arquitectura

La herramienta **Analyze4BPEL** está compuesta por la arquitectura lógica descrita en la Figura 5.7. Como podemos ver, existen tres módulos principales en la herramienta, así como una interacción con elementos externos.

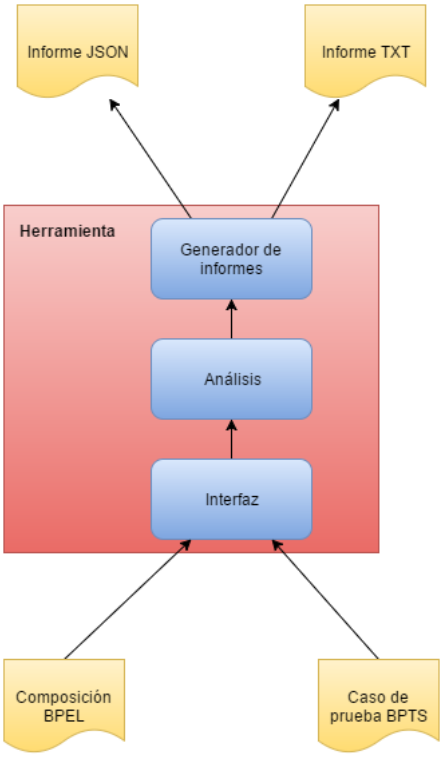

Figura 5.7.: Arquitectura de la herramienta desarrollada

- **Interfaz:** Módulo encargado de gestionar la interacción del usuario con la herramienta. El usuario interaccionará con esta clase para todo mediante una serie de botones, que serán los que ejecutarán el resto de módulos y presentarán los resultados de una manera óptima.
- **Análisis:** Módulo encargado de realizar el análisis de la composición WS-BPEL y el caso de prueba BPTS. El resultado de este análisis será enviado al

módulo generador de informes para generar los informes en el formato buscado.

- **Generador de informes:** Módulo que recibe los resultados del análisis, los formatea y les da una salida apropiada. Esta salida tendrá forma de ficheros con los que el usuario podrá interaccionar para obtener toda la información deseada.

- **Ficheros externos de entrada:**

  Fichero .bpel: Fichero en formato BPEL que representa a la composición que será analizada.

  Fichero .bpts: Fichero en formato BPTS que representa la suite de pruebas que será analizada.

- **Ficheros externos de salida:**

  Ficheros .json: Fichero en formato JSON generado por la herramienta que contiene el informe del análisis de la composición y sus casos de prueba.

  Ficheros .txt: Fichero en formato TXT generado por la herramienta que contiene el informe del análisis de la composición y sus casos de prueba.

La herramienta tiene esta composición modular con el objetivo de facilitar el mantenimiento y ampliación de la herramienta con el tiempo. Además, la modularización de las funciones permiten que estas sean heredadas para su uso en otras herramientas del mismo lenguaje, de esta manera ciertas funciones podrían ser utilizadas de manera independiente sin la necesidad de utilizar todo el proceso implementado para utilizar una de sus funciones.

5.4.2. Diseño de la interfaz

Durante el desarrollo de este TFG se han planteado varias formas de presentación para la herramienta. Tras ver algunos posibles diseños y formas de ejecución, se ha decidido utilizar una interfaz gráfica donde el usuario pueda utilizar la herramienta con la menor interacción posible.
Para la versión final se ha utilizado el diseño planteado en el boceto de la Figura 5.8 puesto que pueden utilizarse todas las funciones desde una única interfaz y con pocas acciones por parte del usuario.

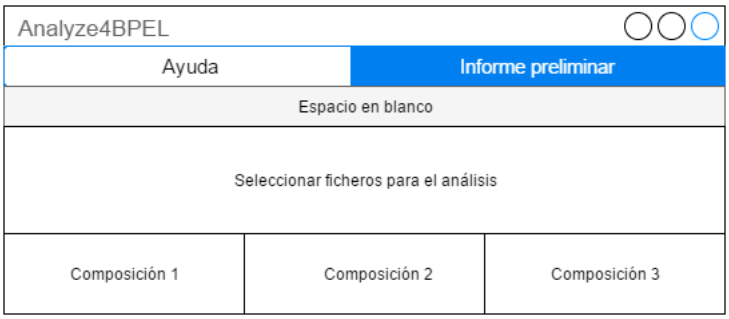

Figura 5.8.: Boceto de la herramienta

5.5. Implementación

A continuación se muestran los detalles de la implementación de los módulos diseñados en la Sección 5.4.

La herramienta **Analyze4BPEL** hace uso de algunas funcionalidades heredadas de otras herramientas. La implementación de estas funciones no será detallada, debido a que solo es necesario obtener el resultado dada una entrada. Puede obtener el código fuente de la aplicación **Analyze4BPEL** en el Anexo D y el código fuente de las clases heredadas en el repositorio de herramientas del grupo de investigación UCASE [20].

Como puede verse en la Sección 5.4, los módulos implementados en la herramienta son:

- Interfaz
- Análisis
- Generador de informes

5.5.1. Módulo de Interfaz

Este módulo se encarga de mostrar la interfaz principal y de manejar todas las interacciones con el usuario.
En la Figura 5.9 podemos ver el diagrama de clases java, donde actúan dos clases: *Main* y *WindowFileSelector*.

La clase *Main* contiene un método encargado de inicializar todos los componentes y dibujar la ventana principal con los botones para la interacción, mientras

que la clase *WindowFileSelector* es la que maneja toda la lógica de la herramienta. Mientras que esta clase actúa como una fachada para el usuario, que solo verá unos sencillos botones, realmente se comunica con todas las funciones de la herramienta mediante una serie de métodos específicos, siendo capaz de ejecutar cada una de sus funciones y generar una serie de mensajes de error o éxito en consecuencia.

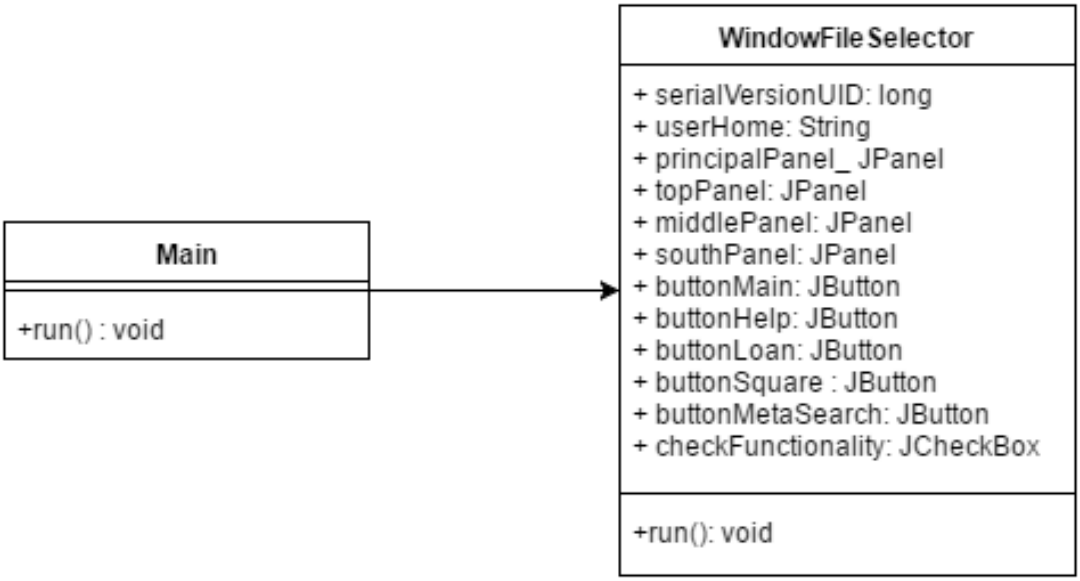

Figura 5.9.: Diagrama de clases del módulo de interfaz

5.5.2. Módulo de Análisis

Aunque funcionan en conjunto, este módulo está dividido en dos submódulos, el de *Información detallada* (el que obtendrá toda la información general de manera indiscriminada) e *Información específica* (el que obtendrá toda la información que ayudará a la obtención de relaciones metamórficas).

En la Figura 5.10 puede verse el diagrama de clases correspondiente al módulo de análisis.

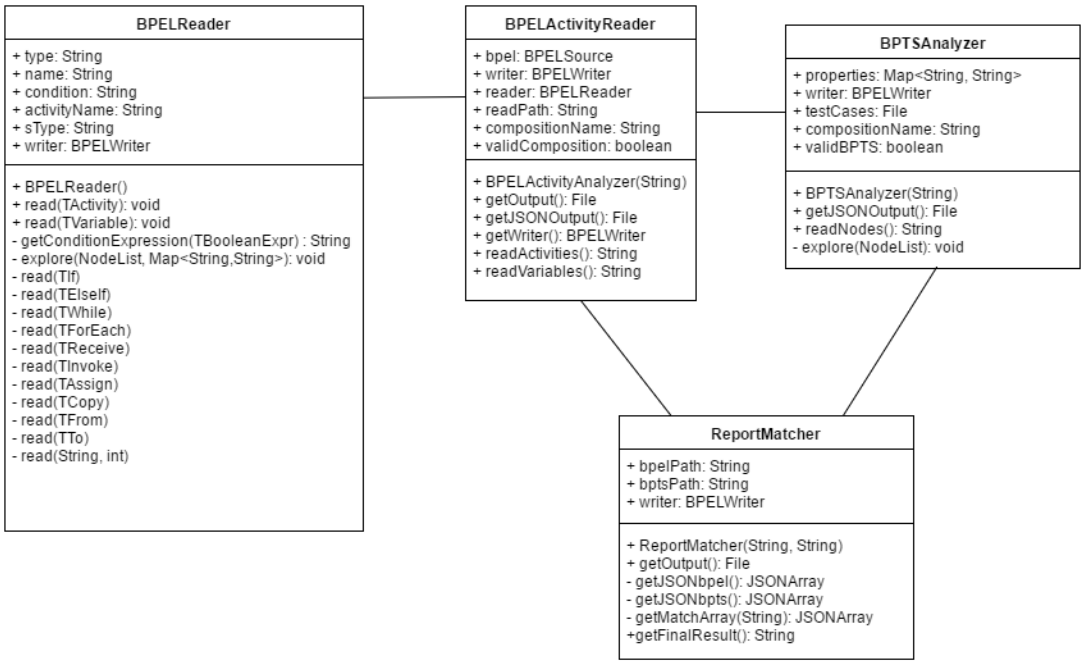

Figura 5.10.: Diagrama de clases del módulo de análisis

5.5.2.1. Submódulo de información detallada

Llamamos información detallada a la información obtenida de manera preliminar que contiene toda la información obtenida de manera indiscriminada.

Este submódulo está compuesto por las clases *BPELReader*, *BPELActivityReader* y *BPTSAnalyzer*. Las dos primeras clases se encargan de obtener la información de los ficheros BPEL, mientras que la tercera clase se encarga de obtener la información de los ficheros BPTS.

El funcionamiento de cada elemento en los siguientes pasos:

1. La clase *BPELActivityReader* obtiene la ruta del fichero .bpel, proporcionado por el módulo de análisis y abre el fichero.

2. Con el fichero abierto, se llama a las funciones *readVariables* y *readActivities* para buscar variables y actividades respectivamente. Estas funciones envían la información contenida en las etiquetas correspondientes (si estuviesen) a la clase *BPELReader*.

   a) Si el elemento recibido es válido, se envía a la función *read* correspondiente. Particularmente la función *read* correspondiente a las actividades enviará cada actividad a otra función específica en función al tipo de actividad que represente. Si el elemento no es válido será ignorado.
   b) De cada elemento se extrae la información que se necesite y se le da un formato adecuado.
   c) La información obtenida es almacenada en un diccionario de datos de la clase *Map*.
   d) Se devuelve toda la información a la clase *BPELActivityReader*.

3. Una vez recibida toda la información, se envía al módulo de generación de informes. (Ver sección 5.5.3)

De esta manera obtenemos toda la información del fichero BPEL, a continuación se muestra el funcionamiento de la clase que obtiene información de los ficheros BPTS:

1. La clase *BPTSReader* obtiene la ruta del fichero .bpts, proporcionado pro el módulo de análisis y abre el fichero.

2. Mediante las funciones *readNodes* y *explore* se explora el árbol de elementos del fichero, que está basado en XML, almacenando toda la información necesaria que se encuentra entre las etiquetas *testCase*, ya que entre ellas podemos encontrar todos los elementos de los casos de prueba.

3. Una vez finalizada la exploración, se envía la información obtenida al módulo de generación de informes. (Ver sección 5.5.3)

De esta manera hemos obtenido toda la información de los ficheros .bpel y .bpts y se ha enviado el módulo de generación de informes para la generación de los informes correspondientes. Sin embargo, esta información se ha obtenido de manera indiscriminada, quedando pendiente un cruce de información donde nos quedemos solo con los elementos que afectan tanto a la composición como a sus casos de prueba. De esto se encarga el *submódulo de información específica*. 5.5.2.2

5.5.2.2. Submódulo de información específica

Este submódulo está compuesto por la clase *ReportMatcher*, encargada de leer toda la información obtenida de la extracción del *submódulo de información detallada* y encontrar toda la información coincidente en la composición BPEL y sus casos de prueba.

Su funcionamiento es el siguiente:

1. Se inicializa con la ruta donde encontrar la información correspondiente.

2. Mediante las funciones *getJSONbpel* y *getJSONbpts* se carga en memoria la información correspondiente.

3. Mediante la función *getMatchArray* se explora cada elemento obtenido en busca de coincidencias.

4. Los elementos coincidentes se almacenan en memoria y se envían al módulo de generación de informes (Ver sección 5.5.3).

De esta manera se ha obtenido toda la información de las composiciones que tienen importancia en los casos de prueba, dejando en segundo plano aquellas que no resultan de interés de estudio.

5.5.3. Módulo de generación de informes

. Este módulo está compuesto por la clase *BPELWriter*, encargada de generar todos los ficheros correspondientes a los informes obtenidos.
En la Figura 5.11 puede verse el diagrama de clases correspondiente al módulo de generación de informes.

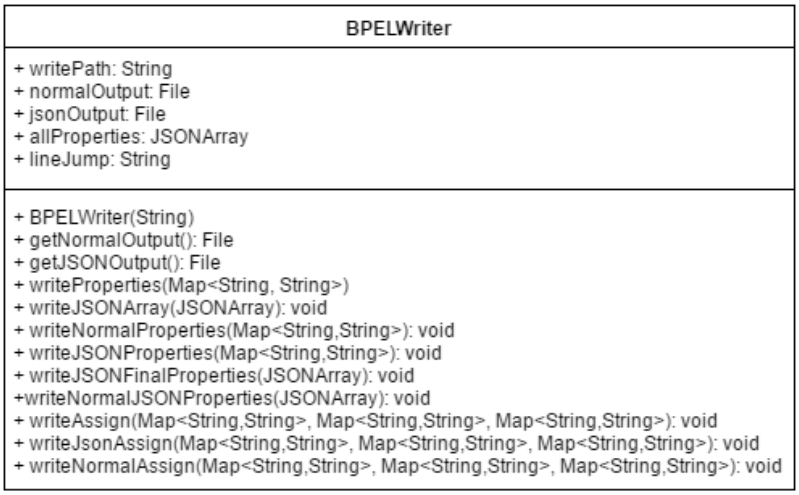

Figura 5.11.: Diagrama de clases del módulo de generación de informes

Aunque este módulo es llamado por los distintos módulos de la herramienta, tiene un funcionamiento genérico:

1. Se inicializa con la ruta donde se escribirán los ficheros.

2. Se enviarán los elementos recibidos a la clase *writeProperties*, encargada de desglosar la información obtenida y de enviar esta a las distintas funciones auxiliares para generar los ficheros de texto y JSON.

Adicionalmente, se han añadido funciones específicas para el tratamiento de las actividades *assign*, ya que estas pueden contener un número indefinido de actividades del lenguaje WS-BPEL. De esta manera, se desacopla esta funcionalidad, obteniendo resultados más rápidos cuando aparecen estas actividades de manera compleja.

5.6. Pruebas y validación

En esta sección se detalla el plan de pruebas seguido para asegurar el correcto funcionamiento de la herramienta desarrollada.

5.6.1. Estrategia y alcance de las pruebas

La herramienta ha ido probándose a la vez que se iba desarrollando. Es decir, cada vez que se ha implementado una función nueva se ha comprobado que su funcionamiento es el correcto y que se integra sin problemas con el resto de funciones ya implementadas.
Durante todo el periodo de desarrollo se han ido comprobando y corrigiendo todos los problemas que han ido apareciendo, consiguiendo así una batería de pruebas unitarias y de integración consistentes al llegar a la versión final de la herramienta.

El alcance de las pruebas realizadas abarca todas las funcionalidades del proyecto, ya sean aquellas públicas disponibles para el usuario final como aquellos subsistemas auxiliares, que en conjunto hacen posible el funcionamiento del sistema.

5.6.2. Entorno de pruebas

Durante el desarrollo, se han realizado la mayoría de las pruebas en un equipo portátil con las siguientes características:

- Sistema Operativo: Linux Mint 17.3 (Versión basada en Ubuntu)
- Entorno de Desarrollo: Eclipse Mars
- Procesador: Intel Core i7-4510U 3.1GHz
- Memoria RAM: 12 GB DDR3

5.6.3. Pruebas Unitarias

El objetivo de estas pruebas consiste en comprobar el correcto funcionamiento de las distintas funcionalidades antes de ser integradas en el sistema final.

El funcionamiento de todas las clases de manera individual ha sido probada manualmente a lo largo de las distintas etapas de desarrollo, sin embargo, para la implementación de ficheros con pruebas unitarias se ha utilizado el framework EVOSuite [12] en su versión **1.0.2**, que permite la implementación automática de pruebas utilizando el framework JUnit [14] cubriendo el 100 % de los casos posibles en el conjunto de pruebas unitarias.
La integración de estas pruebas ha resultado muy reveladora durante el desarrollo de la herramienta y ha permitido descubrir y corregir multitud de errores imperceptibles.

5.6.4. Pruebas de Integración

El objetivo de estas pruebas consiste en encontrar errores en los distintos módulos del sistema implementado comprobando la interacción de los mismos con otros del mismo sistema.

Las pruebas de integración han sido realizadas a la vez que se han ido comunicando los distintos módulos de la herramienta. Se ha comprobado manualmente y con ayuda de las herramientas proporcionadas pro el entorno de desarrollo Eclipse que todos los resultados son los esperados y que no se producen errores durante el intercambio de datos entre los distintos módulos, especialmente entre aquellos que realizan los cálculos necesarios y aquellos que reciben los resultados y los preparan para el usuario final.

5.6.5. Pruebas de Sistema

El objetivo de estas pruebas consiste en verificar que el sistema cumple con todos los requisitos.

Estas pruebas fueron realizadas bajo la interfaz final desarrollada, estableciendo una serie de requisitos basados en las opciones que un usuario final podría realizar. Así mismo, se seleccionaron tres composiciones que cubren todos los casos previstos a la hora de realizar el análisis correspondiente, que son *LoanApproval*, *SquaresSum* y *MetaSearch*.
Puede encontrarse más información acerca de estas composiciones en el Anexo A.

Los pasos seguidos para estas pruebas son:

- **Mostrar instrucciones de uso y autoría**
  - Se pulsa en el botón correspondiente.
  - Se abre una ventana con la información correspondiente.
  - Al pulsar 'Aceptar', vuelve a la ventana principal lista para su uso.
- **Selección de composición auxiliar**
  - Se pulsa en el botón correspondiente.
  - Se abre una ventana de éxito.
  - Se han creado los ficheros en el directorio correspondiente.
  - Al pulsar 'Aceptar', vuelve a la ventana principal lista para su uso.
- **Selección de composición personalizada**
  - Se pulsa en el botón correspondiente.
  - Se seleccionan los ficheros seleccionados, mostrando un error personalizado si no se seleccionan correctamente.
  - Se abre una ventana de éxito.
  - Se han creado los ficheros en el directorio correspondiente.
  - Al pulsar 'Aceptar', vuelve a la ventana principal lista para su uso.
- **Selector de funcionalidad**
  - Todas las pruebas de sistema descritas anteriormente han sido probadas con el botón de funcionalidad en las distintas posiciones, obteniendo resultados positivos.

5.6.6. Pruebas de Aceptación

El objetivo de estas pruebas consiste en verificar que la herramienta está lista para su uso final. Para la realización de estas pruebas, se ha utilizado la herramienta con las distintas composiciones anteriormente mencionadas y se ha intentado obtener de manera manual relaciones metamórficas utilizando solo la información obtenida tras la ejecución del análisis.
Se ha conseguido obtener algunas relaciones metamórficas relevantes con esta información, probando así que el resultado es útil y ahorra recursos con respecto a la obtención de las mismas utilizando las composiciones en su lenguaje.

5.6.7. Validación

Para comprobar que el código implementado cumple con unos requisitos mínimos de calidad, se ha empleado *SonarQube* [19], como puede verse en la Sección 5.6.8.

5.6.8. SonarQube

SonarQube [19] (anteriormente conocido como *Sonar*) es una plataforma que permite analizar código Java de manera que se generan unas métricas capaces de medir la calidad de éste.
Estas métricas permiten ver la calidad del código desde distintas perspectivas entre las que se incluyen fallos y vulnerabilidades (*bugs and vulnerabilities*, malas prácticas (*code smells*), código duplicado (*duplications*) y el tamaño total obtenido tras sumar todos los componentes del proyecto. Además de esto, SonarQube propone un límite donde el código puede considerarse de calidad (*quality gate*), siendo este el límite entre el aprobado y el suspenso.

Para facilitar el proceso de mejora continua, esta herramienta permite realizar análisis a lo largo del tiempo, mostrando mediante gráficas ciertos parámetros como la evolución del código y sus problemas. Estas gráficas permiten revisar todos los parámetros de manera visual, localizando de manera más fácil los problemas y ayudando más a su corrección.

Durante el desarrollo de la herramienta implementada se ha ido revisando todo el código implementado, procurando siempre aprobar el límite de calidad y detectando problemas y vulnerabilidades que de otra manera podrían haber pasado desapercibidas. Finalmente se ha obtenido un proyecto cuyo código cumple con las características de calidad propuestas por esta herramienta. Puede verse en la Figura 5.12 el resultado final del análisis de la herramienta, donde se comprueba el número de problemas es mínimo y se aprueba el límite de calidad.

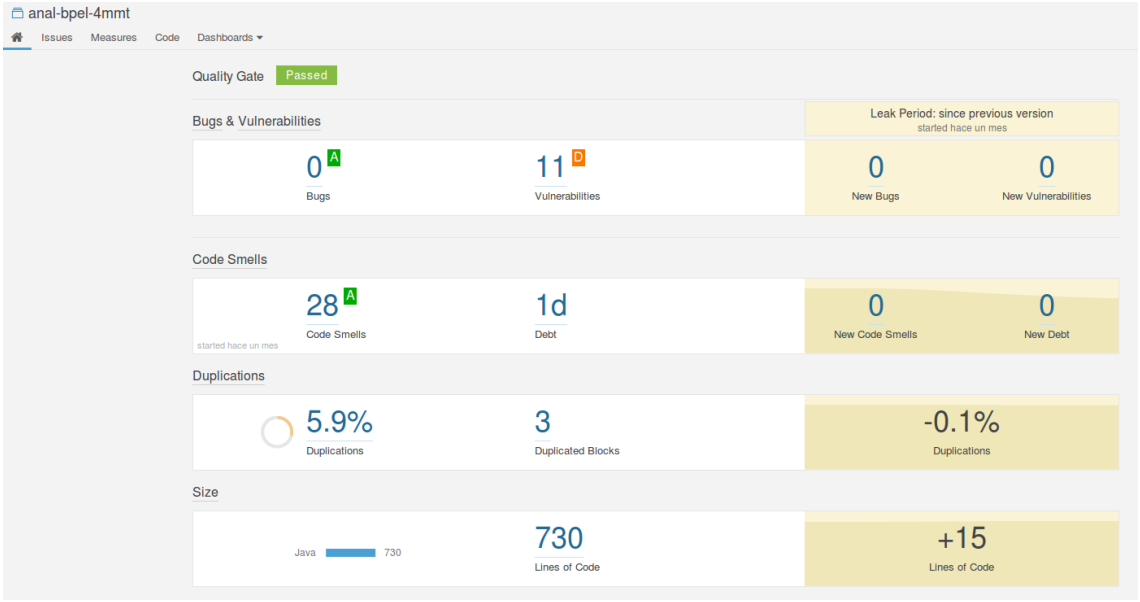

Figura 5.12.: Resultado análisis SonarQube

6. Análisis y Obtención de Información

El proceso de análisis implementado en este TFG es una tarea que se compone de varios pasos. De todos los elementos analizados se pretende obtener una serie de productos que ayudarán a la obtención de RM. En este capítulo se explicará el proceso de análisis paso por paso, así como una explicación centrada en los informes obtenidos tras la aplicación del mismo.

6.1. Proceso de Análisis

Las composiciones WS-BPEL y sus casos de prueba contienen elementos que deben ser tenidos en cuenta a la hora de obtener RM. Aunque todos los elementos de cada uno pueden llevar a la obtención de las mismas, puede observarse como algunos de ellos tienen más frecuencia de uso que otros. Así mismo, se pretende discriminar ciertos atributos de estos elementos, puesto que resultan de mayor interés en la búsqueda de RM.

En el análisis implementado, se realiza una búsqueda en tres pasos:

- Análisis de la composición WS-BPEL.
- Análisis de los casos de prueba.
- Búsqueda de elementos importantes.

A continuación, se detalla cada etapa de este análisis.

6.1.1. Análisis de la composición WS-BPEL

En primer lugar, se pretende extraer información relevante en el formato adecuado de la composición WS-BPEL. A pesar de que todos los elementos de la composición pueden ser objeto de estudio para la obtención de RM, es sabido que ciertos elementos tienden a tener más errores. Es posible que estos elementos contengan errores más claros en el proceso de diseño o implementación de la solución del proceso de

negocio, por lo que deben ser el principal objetivo del análisis a un nivel preliminar.

Estos son todos los elementos de las composiciones WS-BPEL tenidos en cuenta en el análisis implementado acompañados de una justificación de su uso y sus atributos.

6.1.1.1. Variables

Las variables suponen un elemento de gran importancia en las composiciones WS-BPEL. El uso de las mismas tiene una cobertura muy amplia, ya que pueden aparecer en cualquiera de las actividades propias del lenguaje. Pueden tener desde una complejidad muy simple, donde su función se limita a contener un valor sencillo, a una complejidad muy alta, donde se encargan de trabajar con el resultado de complejos procesos.
Las variables son un elemento muy importante en el análisis implementado, ya que su amplia flexibilidad es propensa a contener errores, especialmente en las diferentes asignaciones de sus valor, ya que una sola variable puede variar su contenido tantas veces como se crea necesario en una composición.
A continuación se muestra un ejemplo de una variable en la composición Triángulo.

```
<bpel:variable name="cond0" type="ns1:boolean"></bpel:variable>
```

Los atributos tenidos en cuenta serán el nombre de la variable, para su identificación, y el tipo de elemento que almacena, el cual resultará de utilidad a la hora de comprobar si las asignaciones que se realizan son correctas.

6.1.1.2. Actividades Receive

Las actividades *Receive* son aquellas que reciben una respuesta de otros servicios externos. Aunque no siempre puede comprobarse la validez de estos servicios y suele darse como valores correctos, es necesario tener en cuenta estas llamadas y los valores que manejan, puesto que serán utilizados en la composición.
A continuación se muestra un ejemplo de una actividad Receive en la composición Triángulo.

```
<bpel:receive name="receiveInput" partnerLink="client"
              portType="tns:Triangle"
              operation="process" variable="input"
              createInstance="yes"/>
```

Todos los atributos han sido tenidos en cuenta, ya que podría haber un error en la llamada del servicio provocado por unos valores inadecuados en la llamada del mismo. Se le da especialmente importancia al atributo *variable*, ya que como podemos ver, el valor de respuesta es almacenado en la variable que se indica.

6.1.1.3. Actividades Invoke

Las actividades *Invoke* llaman a servicios externos. A diferencia de las actividades *Receive*, estas actividades no solo reciben respuesta de servidores externos, sino que abren un puerto de intercambio de datos, donde habrá la posibilidad de enviar y recibir datos. Al igual que con las actividades *Receive*, no siempre puede comprobarse la validez de los servicios invocados, por lo que sus resultados suelen tenerse en cuenta como correctos.
Las actividades *Invoke* tienen además la particularidad de ser capaces de crear las variables donde se almacenarán la respuesta del servicio, por lo que debe ser tenida muy en cuenta, ya que las posibles variables creadas pueden tener aparición en la composición sin haber sido previamente declaradas de la misma manera que el resto de variables locales.
Puesto que la composición Triángulo no utiliza ninguna actividad *Invoke*, se muestra a continuación un ejemplo de la actividad correspondiente a la composición *Meta-Search*

```
<bpel:invoke inputVariable="Google_doGoogleSearch_InputVariable"
             name="Google" operation="doGoogleSearch"
             outputVariable="Google_doGoogleSearch_OutputVariable"
             partnerLink="Google" portType="ns1:GoogleSearchPort"/>
```

Todos los atributos han sido tenidos en cuenta, ya que como puede verse, estos trabajan con multitud de elementos. Además del nombre de la actividad y la operación a la que se desea invocar, se indican las variables utilizadas en el proceso, el puerto de entrada y salida que se abre y el servicio externo al que se desea invocar. Todo manejo de variables debe ser tenido en cuenta, ya que puede ser el origen de multitud de errores en la composición, además, podrían darse errores debido a una mala invocación del servicio.

6.1.1.4. Actividades Assign

Las actividades *Assign* son aquellas capaces de asignar uno o más valores a otro conjunto de elementos, como pueden ser las variables. Estas actividades, a su vez, pueden estar compuestas de un número ilímitado de actividades *Copy* acompañadas del mismo número de actividades *From*. Su funcionamiento, aunque muy complejo, resulta muy sencillo de entender: el resultado de la ejecución de la actividad *Copy*, será almacenado según las indicaciones de la actividad *From* asociadas.
Debido a la complejidad que pueden tener las asignaciones, esta actividad debe ser tenida muy en cuenta, ya que es propensa a contener errores en cada uno de los diferentes pasos en su ejecución.
A continuación se muestra un ejemplo de una actividad *Assign* en la composición Triángulo.

```
<bpel:assign validate="no" name="cond0">
    <bpel:copy>
        <bpel:from>
            <![CDATA[$input.payload/tns:a>0 and $input.payload/tns:b>0 ]]>
        </bpel:from>
        <bpel:to variable="cond0"></bpel:to>
    </bpel:copy>
</bpel:assign>
```

En este ejemplo, podemos identificar algunos de los elementos anteriormente indicados. Todos los atributos son tenidos en cuenta, ya que todos influyen en las diferentes asignaciones que pueden hacerse. En este caso, resulta necesario tener en cuenta si se realizará la validación de los valores asignados, así como del valor que se copiará de la actividad *From* a la variable indicada en la actividad *To*.

Aunque este ejemplo puede resultar sencillo de entender, existen otras actividades *Assign* mucho más compleja, donde se llega a trabajar con valores obtenidos desde los distintos servicios externos o con resultados de complejas operaciones, como podemos ver en el ejemplo de la composición *MetaSearch*.

```
<bpel:assign name="GetMaxMSDNNo">
  <bpel:copy>
     <bpel:from>count($MSNSearch_Search_OutputVariable.parameters/
        ns2:Response/ns2:Responses/ns2:SourceResponse/ns2:Results/
        ns2:Result)</bpel:from>
     <bpel:to variable="maxMSN"/>
  </bpel:copy>
  <bpel:copy>
     <bpel:from variable="maxMSN"/>
     <bpel:to part="payload" variable="outputVariable">
        <bpel:query>client:noFromMSN</bpel:query>
     </bpel:to>
  </bpel:copy>
</bpel:assign>
```

6.1.1.5. Actividades If

Las actividades *If*, ampliamente utilizadas en la mayoría de lenguajes de programación utilizados hoy en día, son aquellas que permiten definir un comportamiento condicional específico donde se podrá definir el comportamiento de una, dos o más ramas.
Estas actividades utilizan condiciones propensas a contener errores, especialmente

aquellos cometidos por la persona que codifica la composición. Deben ser tenidas muy en cuenta, ya que a lo largo de sus distintas ramas pueden aparecer otras actividades de cualquier tipo, desde las más simples hasta las más complejas.
Factores como la condición para la ejecución de cada una de sus ramas y el contenido y orden de cada una de ellas debe ser tenido en cuenta a la hora de obtener RM.

Las actividades de este tipo pueden ir acompañadas de actividades *Else*, que contiene la rama a ejecutar si no se cumple la condición propuesta, así como de actividades *ElseIf*, similares a las actividades *Else* pero con la particularidad de que debe cumplirse una nueva condición para que se ejecute su contenido.
A continuación se muestra un ejemplo de una actividad *If* en la composición Triángulo.

```
<bpel:if name="IfCondD">
    <bpel:condition><![CDATA[$input.payload/tns:a=$input.payload/tns:b
                    or $input.payload/tns:b=$input.payload/tns:c
                    or $input.payload/tns:a=$input.payload/tns:c]]>
    </bpel:condition>
    ...
    <bpel:else>
    ...
    </bpel:else>
</bpel:if>
```

Nota: Se ha eliminado el contenido de las ramas, puesto que se compone de distintas asignaciones y en este punto su estudio no es un tema de interés

Todos los atributos han sido en cuenta, así como las condiciones y el contenido de las distintas ramas, que se analizará según el contenido. En el caso de que en vez de una actividad *Else* apareciesen una o más actividades *ElseIf*, todas son tenidas en cuenta junto a su condición particular.

6.1.1.6. Actividades While

Las actividades *While*, también utilizadas en la mayoría de lenguajes de programación de hoy en día, son aquellas en las que se ejecuta una determinada sección un número de veces determinado por la condición propuesta. Es importante tener en cuenta la condición de esta actividad y que se trata de una condición para que su ejecución siga repitiendose, y no una condición de parada.
Puesto que la composición Triángulo no utiliza ninguna actividad *While*, se muestra a continuación un ejemplo de la actividad correspondiente a la composición *Squa-*

resSum 3

```
<while name="PrimeWhile">
    <condition>($counterPrime &lt; $totalNumbers)
           and $output.response/ns0:prime</condition>
    <scope name="PrimeScope">
        <sequence name="PrimeSequence">
               <if name="IsPrimeIf2">
                      <condition>$totalNumbers mod $counterPrime = 0</condition>
                      <assign validate="no" name="PrimeAssign">
                             <copy>
                                    <from>false()</from>
                                    <to>$output.response/ns0:prime</to>
                             </copy>
                      </assign>
               </if>
               <assign validate="no" name="IncreaseCounterPrime">
                      <copy>
                             <from>$counterPrime + 1</from>
                             <to>$counterPrime</to>
                      </copy>
               </assign>
        </sequence>
    </scope>
</while>
```

Todos los atributos han sido tenidos en cuenta, ya que como puede verse, aunque la estructura de la actividad es sencilla y podemos obtener su nombre y condición como únicos elementos propios, es necesario analizar su contenido, ya que puede estar compuesto de multitud de actividades básicas como las que se han tratado anteriormente.

6.1.1.7. Actividades ForEach

Las actividades *ForEach* son muy similares a los elementos *For* de la mayoría de lenguajes utilizados hoy en día. Aunque al igual que la actividad *While* se trata de una actividad cuya ejecución de su contenido se repetirá un número determinado de veces, en esta actividad el número de repeticiones no está condicionado por una condición, sino por un contador determinado por sus atributos.
Puesto que la composición Triángulo no utiliza ninguna actividad *ForEach*, se muestra a continuación un ejemplo de la actividad correspondiente a la composición *squaresSum 3*.

```
<forEach counterName="i" name="CalculateVariance" parallel="yes">
<startCounterValue>1</startCounterValue>
<finalCounterValue>$totalNumbers</finalCounterValue>
<scope isolated="yes" name="ResultLoopBody">
       <sequence>
              <assign name="PartialVariance">
                     <copy>
                            <from>$output.response/ns0:varianceElements/
                            ns0:element[$counter]
                            + $output.response/ns0:variance</from>
                            <to>$output.response/ns0:variance</to>
                     </copy>
              </assign>
              <assign name="IncreaseSumVariance">
                     <copy>
                            <from>-1 + $counter</from>
                            <to>$counter</to>
                     </copy>
              </assign>
       </sequence>
</scope>
</forEach>
```

Todos los atributos han sido tenidos en cuenta debido a la importancia de los mismos a la hora de repetir la ejecución de la sección principal. Con los elementos *startCounterValue* y *finalCounterValue* podemos saber el número de veces que se ejecutará el contenido, concretamente $(finalCounterValue - startCounterValue) + 1$ veces. Además, podemos ver si la ejecución se hará de forma paralela o secuencial, lo cual puede ser origen de errores o comportamientos no esperados que deben ser tenidos en cuenta. Al igual que en ocasiones anteriores, el contenido de la actividad, que puede estar compuesta de una o más actividades, también es analizado.

6.1.2. Análisis de los casos de prueba

Los casos de prueba que acompañan a las composiciones WS-BPEL analizadas se presentan en formato BPTS. Debido a la flexibilidad del mismo y la lógica de la que es dotada para probar los diferentes elementos de la composición WS-BPEL, es importante tener en cuenta todos los elementos que aparecen en los mismos, por lo que el análisis de los casos de prueba se compone de una búsqueda en preorden de elementos relacionados con la composición WS-BPEL.
A continuación, puede verse un ejemplo de un caso de prueba de la composición

SquaresSum 3.

```
<tes:testCase name="OneIteration" basedOn="" abstract="false" vary="false">
  <tes:clientTrack>
    <tes:sendReceive
        service="ap:squaresSumService"
        port="squaresSumPort"
        operation="squaresSumOperation">
      <tes:send fault="false">
        <tes:data>
          <tns:squaresSumRequest>
            <tns:n>1</tns:n>
          </tns:squaresSumRequest>
        </tes:data>
      </tes:send>

      <tes:receive fault="false">
        <tes:condition>
          <tes:expression>tns:squaresSumResponse/tns:sum</tes:expression>
          <tes:value>1</tes:value>
        </tes:condition>
        <tes:condition>
          <tes:expression>tns:squaresSumResponse/tns:variance</tes:expression>
          <tes:value>0.25</tes:value>
        </tes:condition>
        ...
      </tes:receive>
    </tes:sendReceive>
  </tes:clientTrack>
</tes:testCase>
```

Nota: Se ha eliminado parte del contenido del caso de prueba debido a su extensión, ya que en este punto el estudio del contenido particular del mismo no es tema de interés.

Como puede observarse en este ejemplo, los casos de prueba presentan una estructura donde se otorga un valor a la composición y se comprueba una serie de valores esperados. La asignación y obtención de estas expresiones tiene una estructura y etiquetas personalizadas, por lo que se obtendrá toda la información posible tanto de las etiquetas como de su contenido con el objetivo de identificar elementos que se estén teniendo en cuenta de la composición WS-BPEL en el siguiente paso.

6.1.3. Búsqueda de elementos destacados

Llegados a este punto, sin tener en cuenta los formatos en los que han sido producidos, tenemos dos informes: uno que detalla las actividades y sus atributos de la composición WS-BPEL y otro que detalla los elementos utilizados en el caso de prueba analizado.
Aunque estos elementos por si mismo pueden servirnos de utilidad cuando se buscan elementos para la obtención de RM, se busca un informe más reducido, que solo contenga aquella información realmente útil para la obtención de RM cuya aplicación ayude a detectar errores en la composición y sus casos de prueba generando una serie de casos de prueba siguientes más específicos.

Queremos saber qué elementos tienen peso tanto en las composiciones WS-BPEL como en sus casos de prueba, por lo tanto, se genera un informe definitivo mediante los siguientes pasos:

1. Recopilación de resultados de los análisis anteriores.

2. Exploración de cada elemento de los resultados de manera individualizada.

3. Se comprueba si el elemento del caso de prueba correspondiente aparece en alguna de las actividades obtenidas en el informe de la composición WS-BPEL.

4. ¿Se encuentra el elemento del caso de prueba en las actividades obtenidas?

   Sí: Se imprime la información de esta actividad en el informe definitivo.

   No: Si el elemento no aparece en el informe de la composición WS-BPEL, significa que o bien no influye en el comportamiento principal de la composición, como pueden ser las declaraciones típicas del lenguaje, o bien no es un elemento que sea de interés en la obtención de RM, como puede ser una actividad del tipo *Empty*.

En la figura 6.1 se muestra un fragmento del informe generado en este paso tras el análisis de la composición *LoanApproval*.
Nota: Esto es solo un fragmento del informe definitivo, ya que con esta información se pretende explicar el resultado. El informe definitivo es un poco más largo y puede encontrarse junto a los ficheros del proyecto

Como podemos ver, en este informe se le da importancia a la variable *risk* y a las dos comparaciones que se realizan junto a todos los elementos que interfieren en ella. Con esta información, conocemos la existencia de estos elementos y los valores que podrían ayudarnos en la obtención de RM, como puede ser el valor $\leq$ 10,000

```
----------
type - VARIABLE
activityName - risk
----------
type - IF
activityName - IfSmallAmount
condition - ( $request.amount <= 10000 )
else - yes
----------
type - IF
activityName - IfLowRisk
condition - ( $risk.level = 'low' )
else - yes
----------
```

Figura 6.1.: Fragmento de informe definitivo de la composición LoanApproval

utilizado en la primera comparación o el valor = 'low' utilizado en la segunda. Viendo también el nombre de las actividades podemos saber cual es su propósito, en el sentido de que una variable llamada *risk* tendrá una probabilidad alta de encargarse del riego, así como los condicionales llamados *IfSmallAmount* e *IfLowRisk* tendrán una alta probabilidad de comprobar si la cantidad solicitada es pequeña o si el riesgo es bajo respectivamente.

Todos estos elementos han sido obtenidos tras la comparación de los elementos de los casos de prueba con las actividades de la composición WS-BPEL, donde su uso es de gran importancia tanto en la composición como en el caso de prueba que queremos comprobar.

6.2. Resultado del Análisis

El análisis de la composición y sus casos de prueba genera una serie de informes que proporcionan información reveladora que ayudará a obtener RM. Aunque el objetivo principal de este TFG es proveer una herramienta que ayude a automatizar la obtención de las RM, es importante mantener una vía que pueda utilizarse para la obtención manual de estas.
Por este motivo todos los informes generados aparecerán en dos formatos: JSON y texto plano.

6.2.1. Ficheros JSON

Para proveer un método por el cual el resultado de los informes pueda ser automatizado y utilizado por otras aplicaciones, se genera el informe en formato JSON.

Como podemos ver en la Sección 1.6, JSON es uno de los formatos que más facilitan esta tarea.
Todos los informes obtenidos mantienen la misma estructura, siendo esta un *Array de JSON*.
Esta estructura es capaz de contener un número ilimitado de objetos JSON en una estructura vectorial. El informe está codificado de manera que cada objeto JSON se corresponde con un elemento analizado, donde las etiquetas se corresponden con el nombre del atributo obtenido y el valor se corresponde con el valor asignado a ese atributo. Puede verse un ejemplo de esta estructura en la Figura 6.2

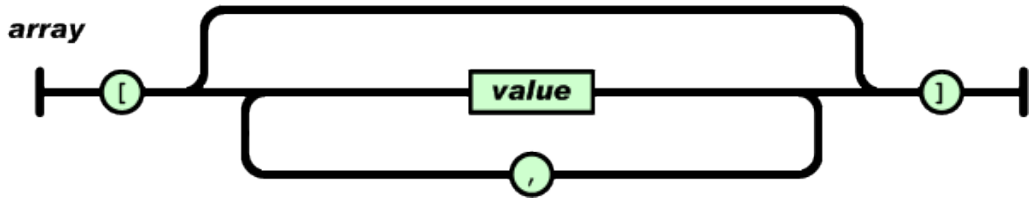

Figura 6.2.: Esquema de Array de JSON

La decisión de utilizar este formato para el informe destinado a la automatización ha sido motivada por el hecho de que la mayoría de los lenguajes de programación utilizados hoy en día proveen librerías capaces de extraer toda la información de esta estructura de datos.
En la Figura 6.3 puede verse un fragmento del informe obtenido tras el análisis inicial de la composición Triángulo.


```
[{"type":"VARIABLE","activityName":"input"},{"type":"VARIABLE","activityName":"cond3"},
{"type":"VARIABLE","activityName":"output"},{"type":"VARIABLE","activityName":"cond2"},
{"type":"VARIABLE","activityName":"cond4"},{"type":"VARIABLE","activityName":"cond5"},
{"type":"VARIABLE","activityName":"cond0"},{"type":"VARIABLE","activityName":"cond1"},
{"portType":"{http:\/\/TriangleProcess}
Triangle","operation":"process","partnerLink":"client","messageExchange":"null","type":"R
{"to":{"variable":"cond0"},"from":{},"activity":
{"type":"ASSIGN","activityName":"cond0"}},{"else":"yes","condition":"$cond0 and
$input.payload\/tns:c>0","type":"IF","activityName":"IfCondA"},{"to":
{"variable":"cond1"},"from":{},"activity":{"type":"ASSIGN","activityName":"cond1"}},
{"to":{"variable":"cond2"},"from":{},"activity":
{"type":"ASSIGN","activityName":"cond2"}},{"to":{"variable":"cond3"},"from":
{},"activity":{"type":"ASSIGN","activityName":"cond3"}},{"to":
{"variable":"cond4"},"from":{},"activity":{"type":"ASSIGN","activityName":"cond4"}},
{"else":"yes","condition":"$cond3 and $cond4","type":"IF","activityName":"IfCondB"},
```


Figura 6.3.: Fragmento de informe en JSON

Como podemos ver, esta estructura no resulta sencilla de analizar para el ojo humano. De hecho, este formato no presentaría ninguna ventaja respecto al lenguaje XML para la persona encargada del análisis, por lo que resulta necesario un informe formateado de manera que los elementos importantes sean reconocidos a simple vista

y cuya lectura no requiera el esfuerzo de decodificar los elementos en código JSON o XML.
En la Sección 6.2.2 se muestra un ejemplo del informe correspondiente en un formato de fácil lectura.

6.2.2. Lenguaje Natural

Como se ha visto anteriormente, además de ayudar a la automatización de la obtención de RM, también se busca proveer un informe que facilite la tarea de obtención manual de relaciones metamórficas. Un informe con la información relevante en un formato de fácil lectura es generado por la herramienta al mismo tiempo que se genera el informe en formato JSON.

Aunque ambos informes contienen exactamente la misma información, este formato permite a la persona encargada identificar aquellos elementos más relevantes de una manera más rápida y con un menor esfuerzo. En la Figura 6.4 puede verse un fragmento del mismo informe visto anteriormente en la Figura 6.3 en el otro formato generado.

En este informe, pueden identificarse los elementos de una manera mucho más sencilla, de manera que cada atributo de cada elemento en particular aparece junto a su valor, como las condiciones o el nombre asignado a los mismos.

```
----------
type - VARIABLE
activityName - cond1
----------
type - IF
activityName - IfCondC
condition - $input.payload/tns:a= $input.payload/tns:b and $input.payload/tns:b=
$input.payload/tns:c
else - yes
----------
activityName - AssignEq
type - ASSIGN
COPY
FROM:
tns:result - 0
TO:
part - payload
variable - output
COPY
FROM:
TO:
part - payload
variable - output
queryLanguage - urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0
bpel:query - tns:result
type - IF
activityName - IfCondD
condition - $input.payload/tns:a=$input.payload/tns:b or $input.payload/tns:b=
$input.payload/tns:c or $input.payload/tns:a=$input.payload/tns:c
else - yes
----------
```

Figura 6.4.: Fragmento de informe en texto plano

7. Conclusiones y Trabajo Futuro

En este capítulo se encuentran las conclusiones de TFG con una valoración del mismo, así como una serie de posibles mejoras cuya implementación será considerada con el objetivo de seguir mejorando el proceso implementado.

7.1. Valoración

El proceso implementado en este TFG surgió con el objetivo de ayudar a automatizar la fase de análisis y obtención de propiedades de las composiciones WS-BPEL para su aplicación en la técnica de prueba metamórfica mediante su integración como uno de los módulos de la arquitectura propuesta anteriormente en [6] por el grupo de investigación UCASE.
Los resultados obtenidos han sido positivos y permiten continuar trabajando en la integración de este sistema en la arquitectura propuesta.

Aunque existe información abundante acerca de la prueba de software en los lenguajes de programación más utilizados, no es así para lo referido a las composiciones WS-BPEL, por lo que se ha dedicado mucho tiempo a la investigación y adquisición de los conocimientos necesarios para poder implementar todo el proceso. Por este motivo, así como, el hecho de que pueda ser utilizado en un entorno de investigación real, hacen que el tiempo invertido sea considerado adecuado.

7.2. Objetivos cumplidos

Todos los objetivos marcados en este TFG han sido cumplidos en su totalidad. Ahora existe una herramienta capaz de obtener información de utilidad a partir de una composición WS-BPEL y sus casos de prueba. Esta herramienta podrá encontrarse en el repositorio del grupo de investigación UCASE y podrá ser utilizada por todo aquel que lo necesite.
A continuación, se muestran los objetivos cumplidos:

- Implementación de un proceso automatizado de parte de la etapa de análisis y obtención de propiedades de la arquitectura propuesta en [6], concretamente, el análisis. Además se genera información para la implementación de propiedades.

- Desarrollo de una herramienta que abstraiga del proceso implementado.
- Integración de la herramienta con otras herramientas que sigan el estilo de desarrollo del grupo de investigación UCASE.
- Diseño de informes formateados para habilitar el tratamiento automático de la información obtenida del análisis.
- Diseño de informes formateados para facilitar la tarea manual del análisis y obtención de propiedades.
- Diseño de una interfaz de uso sencilla y accesible.

7.3. Lecciones aprendidas

El desarrollo de este TFG se ha realizado en el marco de una colaboración académica de dos cursos. Esto ha permitido que se adquieran muchos conocimientos complementarios a la titulación cursada. Además, se ha podido ver de primera mano como funciona un grupo de investigación en toda clase de proyectos, adquiriendo gran cantidad de conocimiento muy positivo para mi formación investigadora y profesional.

A continuación se detallan los conocimientos adquiridos a lo largo del desarrollo de este TFG:

- **Técnicas de prueba de software:** Se ha estudiado y visto en profundidad el concepto de técnicas de prueba de software, de manera que se han visto multitud de técnicas diferentes y como se adaptan según las necesidades de los sistemas y los usuarios. Concretamente se ha centrado el estudio en la técnica de prueba de metamórfica, con el objetivo de comprender totalmente su funcionamiento e implementación para el desarrollo y obtención de relaciones metamórficas a partir de los elementos clave de los sistemas a los que queremos aplicar la técnica.
- **Lenguaje WS-BPEL:** Se ha estudiado el lenguaje WS-BPEL y se han adquirido conocimientos sobre su estructura y su utilización. Los conocimientos adquiridos en este lenguaje han permitido la ejecución de las distintas composiciones implementadas y modificarlas según las necesidades del momento. Este lenguaje está basado en XML, por lo que ha llevado al estudio de este con sus distintos elementos de control, como XMLSchema.

- **BPELUnit Test Suite:** Se ha estudiado esta herramienta y los ficheros de casos de prueba utilizados para la misma. El objetivo ha sido entender por completo el funcionamiento de sus ficheros para integrarlos con el análisis implementado.
- **Servicios web**: Se han adquirido conocimientos de los distintos tipos de servicios web, somo REST y SOAP.
- **Patrones de diseño:** Se han estudiado multitud de patrones de diseño para su aplicación en el desarrollo de la herramienta propuesta.
- **Interfaces gráficas:** Se han obtenido conocimientos sobre el desarrollo de interfaces gráficas para el lenguaje de programación Java.
- **Herramientas de desarrollo:** Se han obtenido conocimientos de herramientas que ayudan al desarrollo de software y su mantenimiento, como Eclipse, Subversion, Git...

Además de esto, se han ampliado los conocimientos del lenguaje de programación Java. Concretamente, se ha estudiado y aplicado el hecho de que ciertas estructuras de datos implementadas en este lenguaje resultan mucho más eficientes que otras. Por ejemplo, en uno de los primeros prototipos de la herramienta, se utilizaban objetos del tipo *HashMap*, que permiten tener una colección de elementos de una manera que se ajusta a la perfección a la que necesitamos, pues bien, en una de las conferencias del SGSOACS (Véase la sección 3.1.10), se habló de que estas estructuras eran muy ineficientes, puesto que para cada elemento que se modifica, la colección de elementos vuelve a ordenarse. Siguiendo las recomendaciones expuestas en dicha conferencia, se cambiaron todos los elementos *HashMap* del código fuente por elementos *LinkedHashMap*, que representan una colección de elementos enlazados, donde el orden queda a disposición del programador. Con este cambio se consiguió que el tiempo de ejecución del análisis se redujera a más de la mitad, haciendo que el proceso sea mucho más óptimo.

Adicionalmente, se han adquirido aptitudes gracias al desarrollo de un artículo para un congreso (Véase la sección 3.1.7) y las múltiples exposiciones del proyecto en sus distintas fases.

7.4. Trabajo futuro

Aunque los objetivos iniciales han sido satisfechos, este trabajo forma parte de un proceso que sigue en desarrollo, por lo que se seguirá mejorando en el futuro para lograr una total integración con el sistema y otras herramientas que lo necesiten.

Otra de las mejoras adicionales de la herramienta consiste en la sustitución de ciertos elementos de la implementación por otros más eficientes y sencillos de usar. Concretamente se sustituirán en el futuro todas las funciones que tienen relación con la búsqueda de elementos en la composición WS-BPEL y los casos de prueba BPTS por plantillas que utilizan la herramienta *XMLStarlet*, la cual permite obtener sentencias XPath de manera automática, cuya integración permitirá analizar todos los elementos en mayor profundidad y mucho más rápido. Una vez implementado un módulo de estas características, se continuaría con la exploración de ficheros CSV y VTL para ofrecer unos resultados fiables acerca de los casos de prueba BPTS que utilizan plantillas en estos formatos. El auge del uso de plantillas hace necesario que el tratamiento de los mismos se haga de forma transparente para el usuario, aumentando así el valor del informe generado.

Respecto a los informes generados por la aplicación, se está trabajando en una mejora que consiste un formato de salida más entendible. Especialmente en el informe destinado al análisis manual, donde se busca aplicar una estructura que facilite aún más la estructura, planteándose la posibilidad de utilizar *texto enriquecido* o un lenguaje como *HTML*, que permita mostrar una interfaz web sencilla e interactiva de manera que se resalten aún más aquellos elementos que sin duda llevarán a obtener relaciones metamórficas.

Finalmente, es necesario seguir aplicando esta herramienta a todas las composiciones posibles, ya que los resultados específicos de cada composición revelan datos que han podido no haber sido tenidos en cuenta en el estudio de las composiciones ya conocidas.

A. Composiciones Estudiadas

Para el desarrollo de este TFG se han estudiado ciertas composiciones desarrolladas en el grupo de investigación UCASE. Estas composiciones han sido utilizadas para entender el comportamiento de las mismas con sus casos de prueba, obteniendo ciertos patrones y elementos útiles para el análisis implementado.

A continuación puede verse una explicación de cada una de ellas. Estas mismas son incluidas en la herramienta para su uso inmediato.

A.1. Gestión de Préstamos (LoanApproval)

La composición *LoanApproval* [21], conocida también como la composición de *Gestión de Préstamos*, consiste en un proceso que permite aprobar préstamos bancarios a partir de una determinada cantidad monetaria, teniendo en cuenta ciertas decisiones externas como la de un aprobador de préstamos y un asesor financiero.

Los siguientes pasos representan el funcionamiento de esta composición:

- Existe un límite en la cantidad de dinero solicitado que influirá directamente en el comportamiento de la composición, en este caso el límite se ha fijado a 10.000 unidades monetarias. Esta cantidad está representada por el elemento req_amount y según se supere o no entrará en acción un agente u otro.
- Si la cantidad solicitada (*req_amount*) es **menor o igual** al límite fijado, que en este caso es 10.000, se hará una consulta al asesor financiero para que nos informe si el riesgo de otorgar el préstamo al cliente es alto o bajo. Su respuesta está representada por el elemento *as_reply* y pueden darse dos casos:
  - El riesgo es bajo (*as_reply = "low"*): al ser poca cantidad con un riesgo bajo, el préstamo es concedido de inmediato, obteniendo como respuesta de la ejecución de la composición "true". Este valor es representado por el elemento *accepted*.
  - El riesgo es alto (*as_reply = "high"*): el asesor ha determinado que el riesgo de otorgar el préstamo al cliente es alto, por lo que se consulta al aprobador de préstamos. Este aprobador tiene la decisión final de otorgar

el préstamo o no, por lo que si su respuesta es positiva (*ap_reply = "true"*), el préstamo será concedido de inmediato (*accepted = "true"*). De igual manera, si la respuesta del aprobador es negativa (*ap_reply = "false"*) el préstamo no será concedido (*accepted = "false"*).

- Si la cantidad solicitada (*req_amount*) es **mayor** que el límite fijado, que en este caso se es 10.000, se consultará directamente al aprobador de préstamos sin pasar antes por el asesor financiero. El aprobador de préstamos tomará la decisión final de otorgar el préstamo o no, por lo que si su respuesta es positiva (*ap_reply = "true"*), el préstamo será concedido de inmediato (*accepted = "true"*). De igual manera, si la respuesta del aprobador es negativa (*ap_reply = "false"*) el préstamo no será concedido (*accepted = "false"*).

Para comprender mejor su funcionamiento, puede verse su diagrama BPMN en la Figura A.1

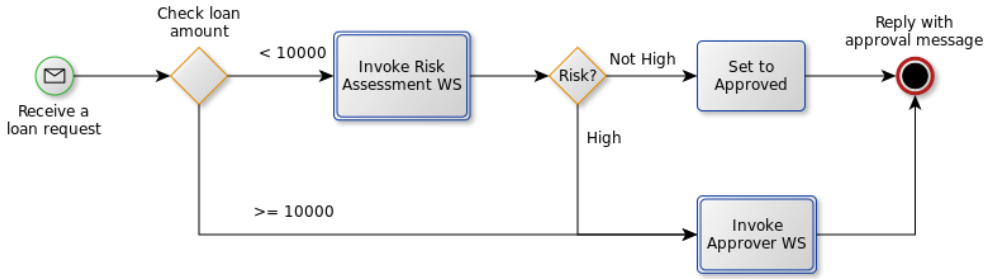

Figura A.1.: Diagrama de la composición LoanApproval

A.2. Suma de Cuadrados (SquaresSum)

Esta composición [23] puede encontrarse en varias versiones con distintos comportamientos, sin embargo, se ha utilizado la versión 3.

Esta composición permite calcular el cuadrado de los entre 0 y 'n', así como la suma total de los mismos, siendo 'n' un número dado. Esta versión permite además obtener la varianza de los números de 0 a 'n' y el resultado de la operación x_2/n de cada número de ese intervalo.

Los siguientes pasos representan el funcionamiento de esta composición:

- La composición recibe un número entero 'n'.

- Se calcula paralelamente el cuadrado de cada número entre 0 y 'n'. Así mismo, se va calculando la suma de todos estos cuadrados y el resultado de la operación x_2/n
- Al igual que en el paso anterior, se calcula de forma paralela la suma de todos los resultados de la operación x_2/n.
- El resultado del paso anterior se resta a la media de los cuadrados calculados para obtener la varianza.
- Se devuelve el resultado.

Para comprender mejor su funcionamiento, puede verse su diagrama BPMN en la Figura A.2

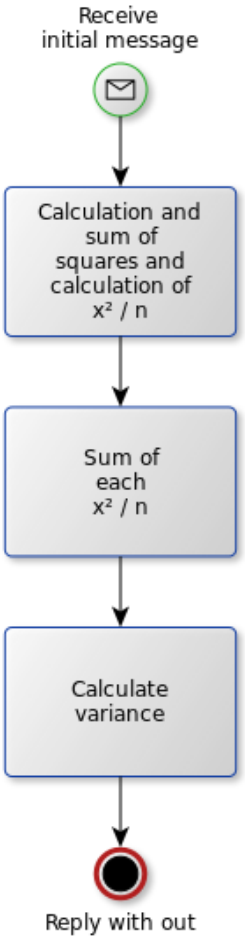

Figura A.2.: Diagrama de la composición SquaresSum

A.3. MetaSearch

Esta composición [22] implementa un motor de 'meta-búsqueda' que realiza diferentes consultas a los motores de Google y MSN. Los resultados obtenidos de estas consultas se combinan y se eliminan los duplicados, obteniendo así los mejores resultados, ya que se aprovechan las diferentes ventajas de estos motores de búsqueda.

Los siguientes pasos representan el funcionamiento de esta composición:

- La composición recibe una consulta.
- Se ejecuta esta consulta en los motores de búsqueda de Google y MSN.
- Si se recibe una respuesta válida de al menos uno de los motores, se ejecutan los siguientes pasos:
  - Se inicializan los contadores a los valores por defecto.
  - Se inicializan ciertas variables internas para enumerar los resultados obtenidos de cada motor de búsqueda.
  - Se combinan los resultados de ambos motores utilizando una estructura de repetición adecuada.
  - Se inicializa el resultado de la respuesta SOAP con los resultados obtenidos.
- Si no se recibe una respuesta válida por parte de alguno de los motores de búsqueda, el resultado se inicializará como un elemento vacío.
- Se devuelve el resultado.

Para comprender mejor su funcionamiento, puede verse su diagrama BPMN en la Figura A.3

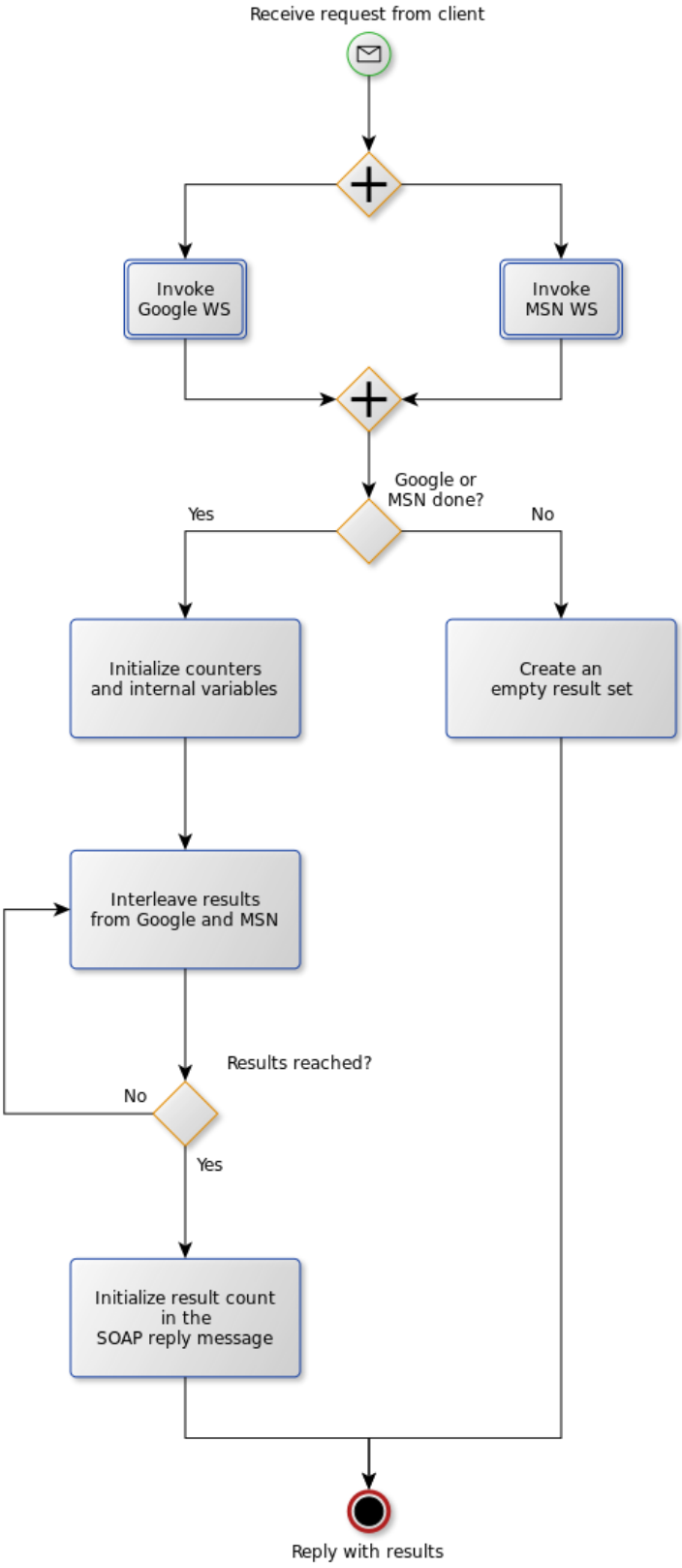

Figura A.3.: Diagrama de la composición MetaSearch

B. Manual de instalación

En este anexo se tratará la instalación del software desarrollado en este TFG, así como todas las dependencias necesarias. Se asegura que esta instalación funcionará en sistemas operativos basados en Ubuntu, aunque se recomienda el uso de Ubuntu 16.04.

Para la instalación de esta aplicación seguiremos los siguientes pasos:

1. Se utilizará el ejecutable .jar para la ejecución de la aplicación. Este mismo se encuentra adjunto en los ficheros de este proyecto. También puede ser descargado a través del repositorio.
2. Colocaremos el fichero en el directorio deseado para su uso.
3. Una vez colocado en el directorio deseado, será necesario darle permisos de ejecución mediante el siguiente comando:

   chmod u+x metamorphic.jar

Es requisito imprescindible tener instalado Java 7. Si tuviese alguna otra versión instalada, será necesario realizar los siguientes pasos:

1. Abrimos una terminal y ejecutamos el siguiente comando:

   sudo apt-get install openjdk-7-jdk

2. Una vez instalado, será necesario cambiar a esta versión como la actual. Ejecutaremos el siguiente comando:

   sudo update-alternatives –config java

3. Aparecerá una nueva pantalla donde deberá seleccionar la opción de Java 7. En el ejemplo de la Figura B.1 seleccionaremos el número 2.
4. Para comprobar que la instalación ha sido correcta, puede ejecutar el siguiente comando:

   java -version

```
Existen 3 opciones para la alternativa java (que provee /usr/bin/java).

  Selección   Ruta                                          Prioridad  Estado
------------------------------------------------------------
  0            /usr/lib/jvm/java-8-oracle/jre/bin/java          1072      modo automático
  1            /usr/lib/jvm/java-6-openjdk-amd64/jre/bin/java   1061      modo manual
* 2            /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java   1071      modo manual
  3            /usr/lib/jvm/java-8-oracle/jre/bin/java          1072      modo manual

Pulse <Intro> para mantener el valor por omisión [*] o pulse un número de selección: 2
```

Figura B.1.: Java Update Alternatives

C. Manual de usuario

Para ejecutar la aplicación, una vez descargada e instalados los requisitos, será necesario abrir una consola en el mismo directorio donde se encuentra *analysis.jar*. En ella, ejecutaremos los siguientes comandos:

java -jar analysis.jar

De esta manera el programa se ejecutará, mostrando la interfaz de la Figura C.1

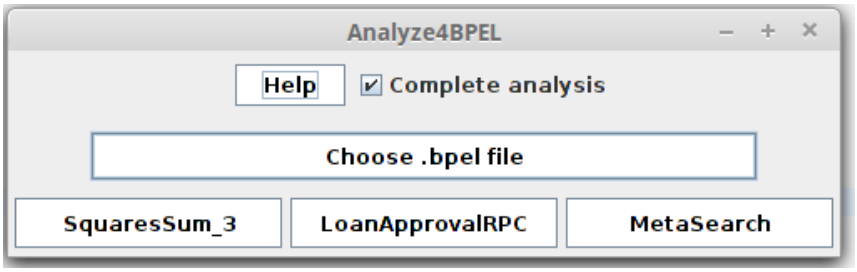

Figura C.1.: Interfaz de Analyzer4BPEL

En la pantalla principal, podemos seleccionar entre dos opciones principales.

1. Ejecutar una de las composiciones de prueba. Estas composiciones están preparadas con un fichero .bpel y su correspondiente .bpts para que pueda ver el funcionamiento de la aplicación en los mismos.

2. Ejecutar una composición personalizada.

Para ejecutar una composición personalizada, seguiremos los siguientes pasos:

1. Pulsamos sobre la opción **Choose .bpel file**.

2. Se abrirá una ventana similar a la Figura C.2. Seleccionamos el fichero .bpel y pulsamos Aceptar.

3. Ahora se abrirá otra ventana similar a la Figura C.3, donde seleccionaremos el fichero .bpts que queremos que acompañe al .bpel.

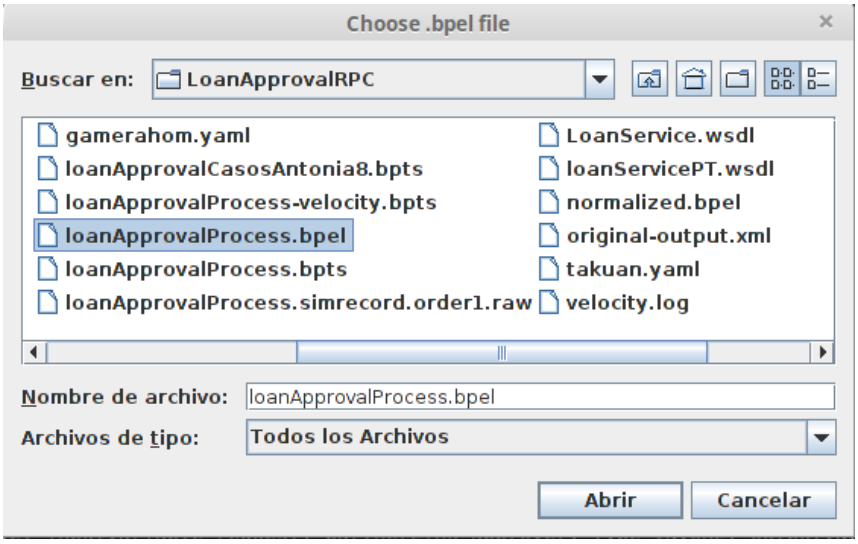

Figura C.2.: Selección de fichero BPEL

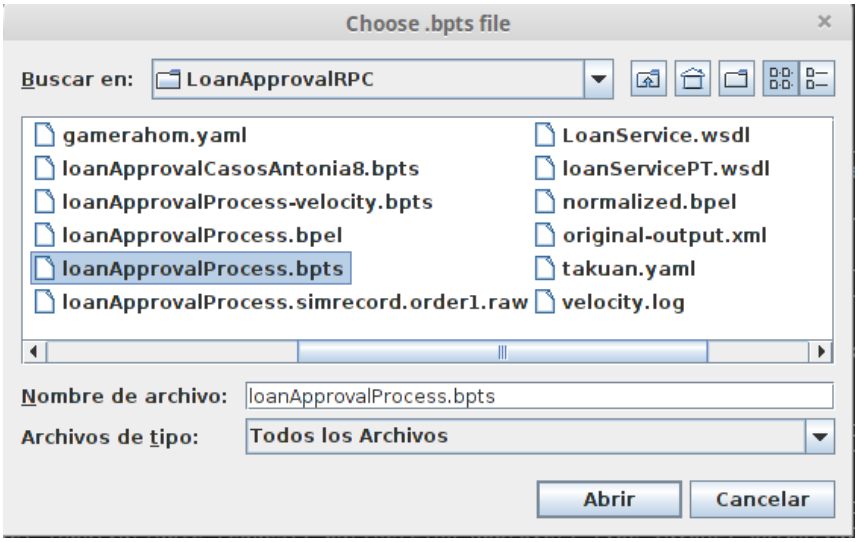

Figura C.3.: Selección de fichero BPTS

4. Tras unos instantes, aparecerá una ventana similar a la Figura C.4, lo cual nos indica que la composición ha sido analizada correctamente. Tras esta ventana se abrirá el informe resultado del análisis, pero puede encontrar el mismo y su versión en formato JSON dentro del directorio */home/user/metamorphic*.

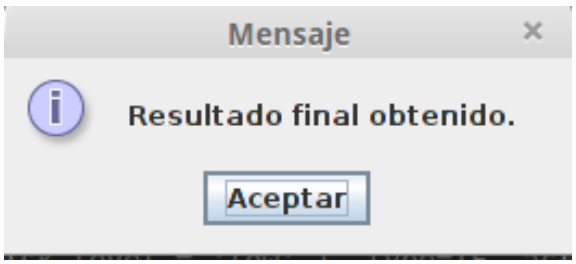

Figura C.4.: Mensaje final de ejecución

D. Manual del desarrollador

En este anexo se explica como descargar el código fuente de este TFG, así como incorporarlo a un entorno de desarrollo para realizar cualquier cambio o ejecutar las pruebas unitarias.

D.1. Descarga del repositorio

Todo el proyecto, incluyendo el código fuente de la herramienta y de este documento, la propia herramienta y las publicaciones relacionadas con la misma desde el siguiente enlace:

https://neptuno.uca.es/git/pfc-kevin-valle

Para ello, será necesario tener instalado el sistema de control de versiones **git**. Una vez instalado, puede ejecutar el siguiente comando desde la terminal en el directorio donde desee descargar los ficheros.

git clone https://neptuno.uca.es/git/pfc-kevin-valle

Pasados unos instantes tendrá una copia del estado actual de este trabajo con la que se podrá trabajar en cualquier aspecto del mismo.

D.2. Importación del proyecto a Eclipse

Para comenzar a trabajar sobre el código fuente, incorporaremos el proyecto a Eclipse. Se recomienda utilizar la versión Eclipse Mars, concretamente el paquete *Eclipse IDE for Java EE Developers*.

Una vez descargado el proyecto siguiendo los pasos de la sección D.1, será necesario descargar e instalar Eclipse. [2]

Es hora de importar el proyecto a Eclipse mediante los siguientes pasos:

1. Pulse en *File*, en el desplegable pulse en *Import...*.

2. Se abrirá una ventana, acuda a la sección *General* y pulse en *Existing Projects into Workspace...*

3. En *Select root directory* pulse en el botón *Browse* y navegue hasta el directorio donde se encuentra descargado el proyecto. Seleccione el directorio *anal-bpel-4mmt*. La ventana debería tener una forma similar a la Figura D.1 en este momento.

4. Pulse el botón *Finish* y tras unos instantes el proyecto habrá sido incorporado a Eclipse correctamente.

A partir de este momento se podrá comenzar a trabajar sobre el código fuente.

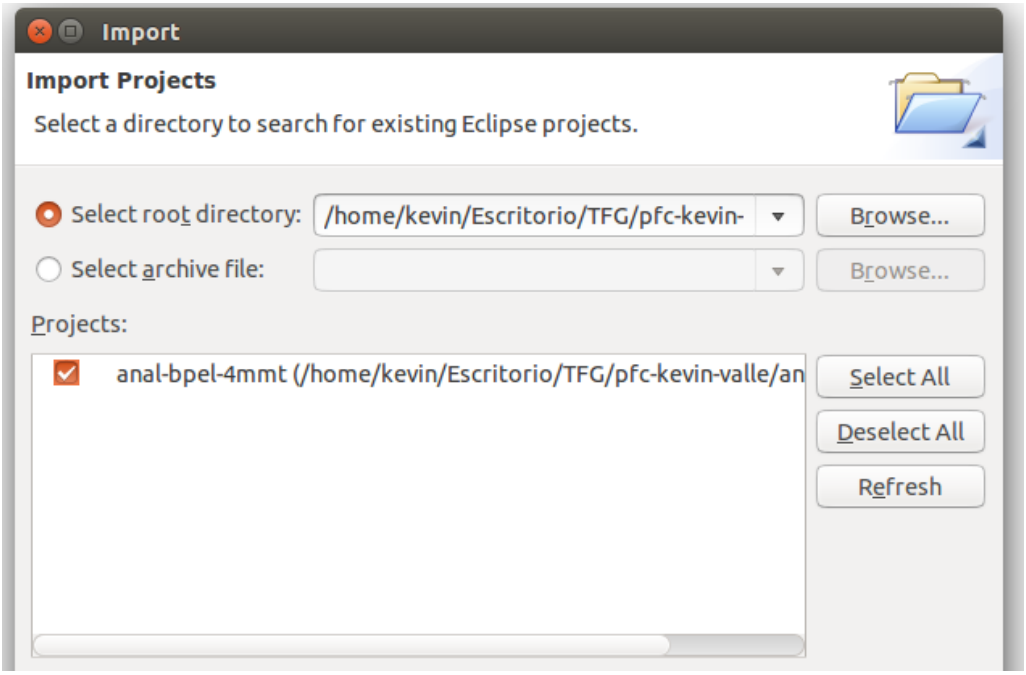

Figura D.1.: Incorporación del proyecto a Eclipse

D.3. Ejecución de pruebas unitarias

Para ejecutar las pruebas unitarias, acuda a Eclipse y en el proyecto, dentro del directorio *test*, haga clic derecho sobre la prueba que desee ejecutar y pulse sobre *Run As*, en el desplegable seleccione *JUnit test*. Una vez hecho esto se abrirá la perspectiva de pruebas y podrá comprobar el resultado de la ejecución de las pruebas implementadas.

En la Figura D.2 puede verse un ejemplo del resultado de la ejecución de uno de los conjuntos de pruebas. El resto de pruebas también son ejecutadas correctamente.

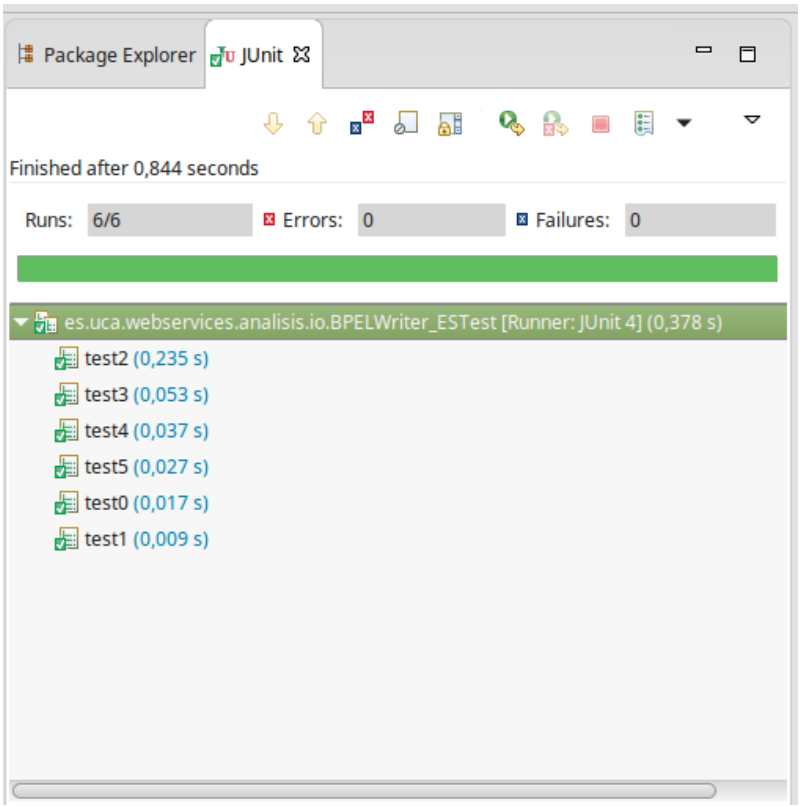

Figura D.2.: Resultado de las pruebas de la clase BPELWriter

D.4. Añadir tratamiento para otras actividades

Para añadir al análisis el tratamiento de otras actividades no contempladas. Será necesario ir a la clase *read* de la clase *BPELReader*.
Una vez ahí, deberá añadir tres líneas que sigan la estructura del resto. Por ejemplo, para contemplar una actividad del tipo 'Nueva', añadiríamos un fragmento similar a la Figura D.3.

A partir de ahora la herramienta estará lista para procesar esa actividad, tan solo será necesario crear la función *read* de ese tipo y dotarla de la lógica que se considere oportuna, siguiendo el patrón del resto de funciones.

```
if(activity instanceof Nueva) {
        read((Nueva) activity);
}
```

Figura D.3.: Nueva actividad para BPELReader

E. GNU Free Documentation License

Version 1.3, 3 November 2008
Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.

<http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "**Title Page**" means, for a printed book, the title page itself, plus such

following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "**publisher**" means any person or entity that distributes copies of the Document to the public.

A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on

the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

Bibliografía

[1] The eclipse fundation: Bpel designer project (2016), http://www.eclipse.org/bpel/

[2] The Eclipse Fundation: Eclipse Main Homepage (2016), http://www.eclipse.org/

[3] Andrews, J.H., Briand, L.C., Labiche, Y.: Is mutation an appropriate tool for testing experiments? In: Proceedings of the 27th International Conference on Software Engineering (ICSE 2005). pp. 402–411. ACM Press (2005)

[4] Apache: Apache Velocity Engine VTL Reference ((accessed Aug 28, 2016)), http://velocity.apache.org/engine/1.7/vtl-reference.html

[5] Baresi, L., Nitto, E.d.: Test and analysis of web services. Springer, Berlin (2007)

[6] Castro-Cabrera, M.d.C., Camacho-Magriñàn, A., Medina-Bulo, I., Palomo-Duarte, M.: Una arquitectura basada en pruebas metamòrficas para composiciones de servicios ws-bpel. In: Actas de las VII Jornadas de Ciencia e Ingeniería de Servicios. pp. 9–22. Servizo de publicacións da Universidade da Coruña, A Coruña, España (Sep 2011)

[7] de Castro-Cabrera, M.d.C., Camacho-Magriñán, A., Medina-Bulo, I.: Aplicación de la técnica de pruebas metamórficas a una composición de servicios: Metasearch pp. 149–154 (2012)

[8] Chen, T.Y., Sc, C., Yiu, S.M.: Metamorphic testing: a new approach for generating next test cases. Tech Rep HKUST-CS98-01 pp. 1–11 (1998)

[9] Chen, T., Huang, D., Tse, T., Zhou, Z.: Case studies on the selection of useful relations in metamorphic testing. Proceedings of the 4th Ibero-American Symposium on Software Engineering and Knowledge Engineering (JIISIC 2004) (Jiisic), 569–583 (2004), http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.63.9943{&}rep=rep1{&}type=pdf

[10] Estero, A., Palomo, F., Medina Bulo, I.: Operadores de mutación para ws-bpel 2.0. In: Actas del III Taller sobre Pruebas en Ingeniería del Software. vol. 2, pp. 13–18 (2008)

[11] García-Domínguez, A.: XMLEye Home Page (2016), https://neptuno.uca.es/redmine/projects/sources-fm/wiki/XMLEye

[12] Gordon Freeser: EvoSuite — Automatic Test Suite Generation for Java ((accessed Jul 24, 2016)), http://www.evosuite.org/

[13] JSON.org: JSON ((accessed Aug 28, 2016)), http://www.json.org

[14] JUnit: JUnit - About ((accessed Jul 24, 2016)), http://junit.org/junit4/

[15] Liñeiro Barea, V.: Herramienta para la generación automática de casos de prueba mediante siembra automática para WS-BPEL 2.0 (April 2016) (2014), http://rodin.uca.es:80/xmlui/handle/10498/16222

[16] 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

[17] Palomo-Duarte, M., García-Domínguez, A., Medina-Bulo, I., Álvarez-Ayllón, A., Santacruz, J.: Takuan: a tool for WS-BPEL composition testing using dynamic invariant generation. In: ICWE. pp. 532–535 (2010)

[18] Pilato, C.M., Collins-Sussman, B., Fitzpatrick, B.W.: Version Control with Subversion. O'Reilly Media, Sebastopol, 2;2nd; edn. (2008)

[19] SonarSource: SonarQube ((accessed Jul 28, 2016)), http://www.sonarqube.org/

[20] UCASE: Redmine ((accessed Jul 28, 2016)), https://neptuno.uca.es/redmine/

[21] UCASE Research Group: LoanApproval - WS-BPEL Composition Repository - Redmine ((accessed Jul 20, 2016)), https://neptuno.uca.es/redmine/projects/wsbpel-comp-repo/wiki/LoanApprovalRPC

[22] UCASE Research Group: MetaSearch - WS-BPEL Composition Repository - Redmine ((accessed Jul 20, 2016)), https://neptuno.uca.es/redmine/projects/wsbpel-comp-repo/wiki/MetaSearch

[23] UCASE Research Group: SquaresSum 2 - WS-BPEL Composition Repository - Redmine ((accessed Jul 20, 2016)), https://neptuno.uca.es/redmine/projects/wsbpel-comp-repo/wiki/SquaresSum_2

[24] Westby, E.J.H.: Git for Teams: A User-Centered Approach to Creating Efficient Workflows in Git. O'Reilly (2015)

[25] Weyuker, E.J.: On Testing Non-testable Programs 25(4), 5–10 (1982)