INVESTIGACIÓN

MÉTODO DE PRUEBAS DE INTEGRACIÓN EN ARQUITECTURAS ORIENTADAS A SERVICIOS

METHOD OF INTEGRATION TEST SERVICE ORIENTED-ARCHITECTURES

MSc.Nelson Beltrán Galvís[a], Ing. Carlos René Angarita Sanguino[b] MSc. Claudia Yamile Gomez Llanez[c]

[a]Docente Planta, GIDIS - UFPS Grupo de Investigación de Ingeniería de Software. Avenida Gran Colombia No. 12E-96 Barrio Colsag, San José de Cúcuta, Colombia, nelsonbeltran@ufps.edu.co

[b]Docente Catedrático, GIDIS - UFPS Grupo de Investigación de Ingeniería de Software. Avenida Gran Colombia No. 12E-96 Barrio Colsag, San José de Cúcuta, Colombia, carlosreneas@ufps.edu.co-crangarita@gmail.com

[c]Docente Catedrático, GIDIS - UFPS Grupo de Investigación de Ingeniería de Software. Avenida Gran Colombia No. 12E-96 Barrio Colsag, San José de Cúcuta, Colombia, claudiaygomez@ufps.edu.co

Fecha de recepción: 11-06-2016 **Fecha de aprobación:** 02-12-2016

Resumen: Las nuevas tendencias en los avances tecnológicos apuntan a la integración de sistemas y gran parte de las empresas lo están haciendo o desean hacerlo; con base en esto, las empresas contratan expertos informáticos que les indiquen las estrategias a seguir para la implementación de lo último en tecnología, conduciéndolos irremediablemente a los Web Services o sistemas con Arquitecturas Orientadas a Servicios (SOA), soportados en la hipótesis de mantener interoperabilidad entre los sistemas actuales de la organización. Los SOA dentro de su metodología de desarrollo cuentan con una etapa de prueba, donde se verifica la calidad de los distintos componentes desarrollados, esta etapa es abordada con procesos de pruebas de arquitecturas tradicionales, enfocando sus esfuerzos en asegurar la calidad en el nivel de unidades y no en el nivel de integración, entendiendo por integración al proceso de consumo de las interfaces de los servicios que son expuestos. En el siguiente artículo se propone un método para el diseño y ejecución de pruebas de integración en arquitecturas orientadas a servicios, con un caso de estudio donde se aplica el método descrito, con los resultados de la aplicación del método descrito en un proyecto en la Universidad Simón Bolívar sede Cúcuta, Colombia.

Palabras clave: Pruebas de Software, Pruebas de Integración, Arquitecturas Orientadas a Servicios, Web Service.

Abstract: Pruebas de Software, Pruebas de Integración, Arquitecturas Orientadas a Servicios, Web Service.

Keywords: Software Test, Integration Test, Service Oriented Architecture, Web Service.

1. INTRODUCCIÓN

Las empresas actualmente cuentan con un conjunto de aplicaciones que pueden ser llamadas heterogéneas, ya que fueron desarrolladas con diversas tecnologías en diversos tiempos, a través de un crecimiento evolutivo del negocio y de los sistemas en sí. La tendencia actual apunta a que los datos y sistemas deban estar integrados para consolidar la información empresarial. A lo anterior se une el hecho de que no solo nos interesa interactuar con nuestros sistemas, sino que la interacción se debe dar con sistemas de terceros, los cuales potencian nuestro negocio. Un ejemplo común es el proceso de pagos en línea, donde las empresas utilizan servicios de terceros para conectar con plataformas bancarias y obtener resultados de confirmación o rechazo.

Las Arquitecturas Orientadas a Servicios (SOA) han tomado fuerza para dar solución a lo anterior, soportados en la hipótesis de mantener interoperabilidad entre los sistemas actuales de la organización, a través del uso generalmente de Servicios Web.

Los SOA (Service Oriented Architecture) dentro de su metodología de desarrollo cuentan con una etapa de pruebas, donde se verifica la calidad de los distintos componentes desarrollados. Aunque las pruebas de software están determinadas por la naturaleza de las aplicaciones, es importante saber que las mismas no están claramente definidas en SOA, debido a la poca madurez, así como las pocas herramientas disponibles, por lo que esta etapa es abordada con procesos de pruebas de arquitecturas tradicionales, enfocando sus esfuerzos en asegurar la calidad en el nivel de unidades y no en el nivel de integración. Además las pruebas sobre aplicaciones tradicionales se centran en la lógica del negocio, utilizando la interfaz del usuario, el equipo de desarrollo puede probar los defectos en el sistema a nivel de escritorio, dado que los sistemas basados en interfaz gráfica de usuario no son demasiado dependientes unos de otros, los defectos se pueden encontrar con mayor facilidad, contenerlos, aislarlos, y repararlos, pero en SOA, no se puede esperar un escenario en el que las aplicaciones son desarrolladas por un mismo equipo, los servicios se basan en tecnologías heterogéneas y el uso de componentes distribuidos para crear los procesos de negocio de la organización, además los servicios no tienen interfaz de usuario. Basado en esto debemos identificar dos escenario en las arquitecturas SOA, el primero donde tenemos control de las partes que componen el servicio ya que tenemos acceso a los componentes porque estos son de nuestra empresa y el segundo cuando los

servicios pertenecen a otro equipo por lo tanto solo tenemos acceso a la interfaz.

La falta de procesos claros para el diseño de pruebas en proyectos de software con arquitecturas orientadas a servicios, tiende a que los errores sean descubiertos hasta la etapa de implantación, fase donde el costo del cambio es más alto, reflejando un incremento en el valor total del proyecto.

Para apoyar en el proceso de aplicación de la pruebas de integración se propone un método de pruebas de Integración para desarrollos de software con Arquitecturas Orientadas a Servicios que comprende cuatro fases que van desde el conocimiento del entorno de desarrollo hasta la ejecución de las pruebas. También se exponen los resultados de la aplicación del método en un proyecto WebSimon para la gestión de información de estudiantes en la Universidad Simón Bolívar sede Cúcuta.

2. METODOLOGÍA

Este proyecto considera ser abordado a través de una metodología cualitativa, debido a que se formula un método de pruebas basado en el funcionamiento del proceso de desarrollo de software y en un conjunto de procedimientos aplicables a la disciplina de las pruebas de software. Todo se basa en documentación y observación de los distintos proyectos que se desarrollan, del comportamiento de los participantes, así como de los resultados obtenidos en los mismos.

Para realizar este trabajo se inicia buscando las fuentes bibliográficas que contienen información del objeto de estudio, luego se procede a tomar las definiciones más relevantes de autores que se encuentren en el área de la ingeniería del software; se realiza una documentación de las técnicas de pruebas de software en servicios web y pruebas de integración, y se elabora un estado del arte con respecto al objeto de estudio, lo que sirve de aporte al marco teórico de esta investigación.

Consecutivamente se procede a diseñar cada una de las fases que contiene el método propuesto y así mismo se realiza la validación del método a través de un ejemplo.

En esta sección se enmarcan algunas bases conceptuales y teóricas de las pruebas de software y los servicios Web.

2.1 PRUEBAS DE SOFTWARE

Las pruebas de software consisten en verificar el comportamiento de un programa dinámicamente a través de un grupo finito de casos de prueba, debidamente seleccionados del ámbito de ejecuciones infinito, en relación al comportamiento esperado. Hacer pruebas es una actividad que tiene el objetico de evaluar y mejorar la calidad del producto, identificando defectos y problemas (SWEBOK, 2004).

Las Pruebas de Software se han convertido en una actividad de gran importancia dentro de la Ingeniería de Software, elemento que nos brinda la oportunidad detectar fallos y evaluar las características del software. Las pruebas han evolucionado a lo largo del tiempo, han pasado de ser una simple etapa en el proceso de desarrollo a constituirse en un conjunto de etapas que controlan la duración del ciclo de vida, la calidad y la confiabilidad del software desarrollado.

Actualmente con el incremento del desarrollo de productos de software, y las tendencias de aseguramiento de la calidad en los mismos, se ha considerado la importancia de las Pruebas de Software como proceso de aseguramiento de la calidad dentro de la Ingeniería de Software,

tanto así que la fase de pruebas es una de las más costosas del ciclo de vida software. Un caso concreto son los métodos XP, Scrum y OpenUP que aplican ciclos de desarrollo iterativos y que se basan en los principios definidos por el modelo de desarrollo guiado por pruebas (TDD). Las pruebas son una disciplina compleja, conformada por muchos elementos técnicos y de gestión, los cuales son definidos en (Gelvez, 2010).

Ámbito de las pruebas: corresponde a la etapa o alcance de las pruebas, unitarias, de componentes, de integración, de sistema, de validación, de aceptación (Paul Baker, 2008).

Tipos de prueba: de acuerdo a las dimensiones que desean evaluar se clasifica en funcionales, de usabilidad, confiabilidad, integridad, rendimiento, infraestructura y regresión (Ahmad K. Shuja, 2007).

Ciclos de aplicación: en (Paul Baker, 2008) se tratan como un proyecto relacionado con el proceso de desarrollo, definiendo las fase de requisitos, diseño, especificación, implementación, validación, ejecución y evaluación.

Proceso de pruebas: hay diversos procesos, en (Mark Utting, 2007) se describen los mas usados, como prueba manual, grabación, basadas en secuencia de comandos, automatizadas, basadas en modelos.

2.2 SERVICIOS WEB Y SOA

Service Oriented Architecture (SOA) según (Krafzid, Banke, & Slama, 2005) es un estilo de Arquitectura de Software basado en la definición de servicios reutilizables con interfaces públicas bien definidas, donde proveedores y consumidores de servicios interactúan desacopladamente para realizar los procesos del negocio, y donde los servicios se componen en secuencias definidas para realizar los procesos de negocio (orquestación, coreografía). En (Gelvez, 2010) es una arquitectura de software que define la utilización de servicios, que son componentes de alto nivel que utilizan servicios web, para dar soporte a los requerimientos de software del usuario. Como meta principal se plantea la reusabilidad e interoperabilidad de las aplicaciones obtenidas, mediante la definición de servicios que puedan ser reutilizados por la Organización y fuera de ésta. La mayoría de las definiciones de SOA identifican la utilización de Servicios Web (empleando SOAP y WSDL) en su implementación.

En (Krafzid, Banke, & Slama, 2005) se definen cuatro abstracciones básicas para el estilo SOA: servicios, aplicaciones front-end, repositorio de servicios y bus de servicios. El paradigma descubrir-ligar-invocar es la base del enfoque para el desacoplamiento de servicios, donde los productores de servicios los registran en el repositorio, los consumidores de servicios los buscan en el repositorio, y si existen obtienen una referencia para realizar el ligamiento e invocarlos.

Según (Krafzid, Banke, & Slama, 2005) un servicio consiste en una implementación que provee lógica de negocio y datos, un contrato de servicio que especifica las interfaz que expone físicamente la funcionalidad. Según (Gelvez, 2010) un servicio es una pieza de código independiente que en conjunto representan un entorno de aplicación. Los servicios poseen unas características únicas que les permiten formar parte de una arquitectura orientada a servicios, son autónomos e independientes de otros servicios y permiten crear funcionalidades aisladas del negocio (representan la lógica de una unidad de negocio).

Los servicios representan grupos lógicos de operaciones relacionadas con algún concepto del negocio. Las aplicaciones front-end consumen los servicios y/o los exponen, el repositorio de servicios almacena los contratos de servicios, y el bus de servicios interconecta las aplicaciones front-end y los servicios. Además los servicios pueden clasificarse según su propósito en servicios orientados a procesos que realizan los procesos de negocio, servicios intermediarios, básicos y públicos empresariales (B2B). Aparecen dos nuevas capas de abstracción: procesos de negocio y servicios, en las que se modelan los tipos de servicios y su composición.

Se hace necesario contar con una metodología para desarrollo de software con enfoque SOA que permita realizar un diseño acorde a los requerimientos planteados. Una metodología para desarrollo con enfoque SOA se propone en (Delgado & Garcia, 2010).

2.3 PRUEBAS DE INTEGRACION DE SERVICIOS

En esta arquitectura este tipo de pruebas se centran en las interfaces de los servicios. El objetivo es evaluar la interfaz y el intercambio de información entre los servicios, estas aseguran que la comunicación entre servicios funciona correctamente consumiendo la aplicación y analizando los resultados sin validar su comportamiento interno. En (Gelvez, 2010) se determina el cumplimiento de la interfaz de los servicios con los términos de los estándares, los formatos y la validación de los datos.

En (Torry Harris Bussiness Solutions, 2007) se menciona algunos aspectos importantes:

- Determinar las variaciones de los datos utilizados para probar el servicio.
- Relacionar los casos de prueba con los elementos que incluyan servicios involucrados y las operaciones específicas, de esta forma se puede determinar que pruebas repetir si el servicio cambia.

En (IBM, 2015) se plantean los inconvenientes de las aplicaciones de integración y definen una mejor estrategia, basados en el hecho de que en un SOA la lógica de las aplicaciones está ubicada dentro del nivel intermedio, funcionando con un número indeterminado de tecnologías, residiendo fuera del departamento o incluso fuera de la empresa. Por este motivo, las pruebas de extremo a extremo, incluso en el entorno de prueba, dependen de terceros. Si un sistema externo crítico no se encuentra disponible, tendrá el potencial de interrumpir el ciclo de aplicación de pruebas en caso de que el equipo no esté preparado.

Para poder estar preparados, la literatura incluye el concepto de falsos servicios, como una estrategia que permita a los testers garantizar la disponibilidad de la funcionalidad, con el solo uso de la interfaz del servicio.

Los falsos servicios ayudan a los equipos a reducir la dependencia respecto de los involucrados cuando se ejecuta la prueba de integración. El equipo puede crear servicios que se comportan de manera similar a los servicios reales. Estos servicios son generados usando los esquemas reales (WSDL y compatibles) del servicio original. Luego se dota al falso servicio de un conjunto de respuestas, que se utiliza para responder a la solicitud del falso servicio. Las respuestas pueden ser aleatorias, consistir en una operación por turnos o basarse en ciertas reglas o directivas.

Los falsos servicios se utilizan si el servicio real no está disponible, esto garantiza que la aplicación de pruebas no se vea afectada por retardos en la implementación de los servicios. De igual forma estos falsos servicios ayudan a generar un ambiente de pruebas más controlado.

En (IBM, 2015) proponen mantener suites de pruebas de integración automatizadas y guiarse por un marco de pruebas de servicios definido por ellos.

Dentro del marco de pruebas se definen algunas capacidades comunes que se deben desarrollar:

- La capacidad de producir agentes de prueba en ausencia de la interfaz de usuario de una aplicación
- Generación de mensajes de prueba, basados en la descripción del servicio (WSDL)
- Variación de datos de entrada, usando una tabla de datos
- Scripts de desmontaje y configuración de datos
- Datos de salida de informes de pruebas
- Definición de resultados esperados
- Ejecución de pruebas en cada nivel integrado de la pila (por lo general, a través de un entorno de prueba unitaria)
- Emulación de servicios externos (falsos)
- Inspección y validación de mensajes de servicio de aplicaciones consumidoras
- Envío de múltiples mensajes de prueba a través de subprocesos separados

En (Facundo, 2015) definen un proceso de automatización de las pruebas de integración a través de la creación de un framework para la automatización de pruebas, sin necesidad de crear los servicios finales.

Todo el proceso de las pruebas de integración se basa en las interfaces que interconectan a los servicios, por lo tanto es necesario realizar el análisis de los elementos que componen.

3. RESULTADOS

METODO DE PRUEBA DE INTEGRACIÓN PARA ARQUITECTURA ORIENTADAS A SERVICIOS

El método es un conjunto de pasos que genera uno o varios resultados; también se define como un procedimiento que presenta un inicio y un fin.

El desarrollo de este método permite a las personas encargadas de realizar pruebas en arquitecturas orientadas a servicios, tener una guía para el diseño y ejecución de pruebas de integración.

Se recomienda a la persona que desee implementar este método seguir las siguientes recomendaciones:

Seguir cada fase de forma ordenada.

Mantener comunicación con el equipo de desarrollo sobre las actividades realizadas.

En la **¡Error! No se encuentra el origen de la referencia.** se presenta el desglose del método propuesto el cual se divide en cuatro fases: 1) Conocimiento global del entorno de desarrollo, 2) Diseño de los casos de prueba de integración de los servicios, 3) Implementación de la prueba de integración de servicios, 4) Ejecución de la prueba y análisis de resultados y las actividades correspondientes a cada fase.

En esta sección se enmarcan algunas bases conceptuales y teóricas de las pruebas de software y los servicios Web.

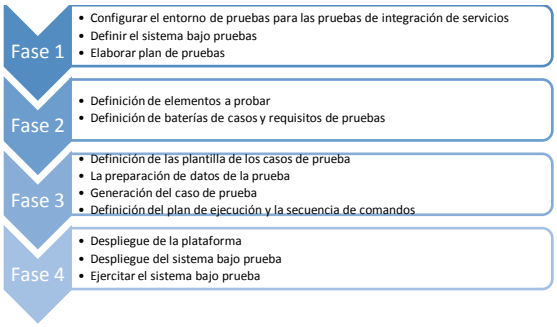

Figura 1. Fases del método de pruebas
Fuente: Elaboración propia.

A continuación se describirán cada una de las fases que componen al método para pruebas de integración para arquitecturas orientadas a servicios.

3.1 Fase 1 - Conocimiento del entorno de desarrollo

En esta fase el probador necesita conocer las características del entorno de desarrollo como son el código, la documentación y los requisitos del sistema. Para lo cual se han diseñado tres actividades.

3.1.1 Configurar el entorno de pruebas para las pruebas de integración de servicios

En esta primera actividad se deben configurar los elementos que serán utilizados en el entorno de pruebas. Por la naturaleza de las pruebas, la técnica para la implementación de las pruebas de integración se basara en el concepto de pruebas de caja negra, utilizando las interfaces de los servicios para crear plantillas de pruebas y realizar inyección de datos sobre los mismos.

Dentro de esta actividad se deben tener varios ítems para poner a punto el entorno donde se ejercitara el software bajo pruebas:

- Configurar las herramientas que serán utilizadas para este proceso
- Configurar el espacio de trabajo, definiendo las estructuras de almacenamiento para organizar los artefactos de las pruebas.
- Verificar rutas para acceso a los elementos del software bajo prueba.
- Definir los elementos necesarios para iniciar el sistema bajo pruebas (contenedores de la aplicación, bases de datos, servidores) de ser necesario y si los servicios a utilizar estuviesen bajo el control del mismo equipo de trabajo. En caso de que los servicios sean de terceros se debe definir si los WSDL de las interfaces a las cuales se accederá se almacenaran local o externamente.
- Automatizar la configuración del entorno de pruebas con el uso de comandos, para lo cual se utiliza la herramienta Apache Ant[1], que igualmente debe ser configurada.

3.1.2 Definir el sistema bajo pruebas

En esta actividad el probador debe definir el alcance de las pruebas, para lo cual debe conocer los requerimientos del sistema y conocer la documentación de los mismos, basándose por ejemplo en historias de usuarios o casos de uso y demás elementos de la metodología. Igualmente se debe tener un conocimiento claro de la metodología a utilizar para poder determinar la forma y momento en que los servicios son consumidos.

Para poder determinar el tamaño del software bajo prueba, se deben definir los servicios que van a ser consumidos por la aplicación. Para este proceso se propone la Figura 2 que contiene la plantilla de Historia del Servicio, la misma se utiliza para cada servicio y se define cada método y requerimiento cubierto.

[1] Apache Ant es una herramienta usada en programación para la realización de tareas mecánicas y repetitivas, normalmente durante la fase de compilación y construcción.

Información del Servicio

Identificación				
Nombre del Servicio				
Descripción del Servicio				
Naturaleza		Interno		Externo
WSDL				
Documento Guía				

Listado de Métodos

Identificación						
Nombre						
Requerimiento						
Criticidad		Alto		Medio		Bajo
Parámetros de entrada						
Nombre	Tipo	Observación				
Parámetros de salida						
Nombre	Tipo	Observación				

Figura 2. Plantilla de historia de servicio
Fuente: Elaboración propia.

Algo importante que el probador debe incluir son los parámetros de entrada y parámetros de salida de cada método, los cuales deben ser abstraídos de la WSDL del servicio. Es importante identificar los métodos a través de la WSDL, ya que teóricamente, si el servicio al interior cambia, la WSDL debe mantenerse igual.

3.1.3 Elaborar plan de pruebas

Esta actividad tiene como objetivo la creación del plan de prueba, que defina la programación de las actividades de prueba. Este sistema identifica los servicios que serán probados, las tareas a ejecutar, los resultados esperados, los responsables y requerimientos de la plataforma.

3.2 Fase 2 - Diseño y especificación de los casos de prueba de integración de los servicios

En esta fase el probador define los casos de prueba juntos con los datos de prueba necesarios para ejercitar el software bajo prueba. De esta fase se deben definir las plantillas de los casos de prueba. El probador debe verificar cada servicio para determinar los datos de entrada que son solicitados por el servicio y definir las respuestas esperadas para poder validar la veracidad de la prueba. El probador debe definir los falsos servicios que se crearan para dar continuidad al proceso de pruebas en llegado caso que los servicios no se encuentren disponibles. Esta fase se divide en dos actividades.

3.2.1 Definición de elementos a probar

Se toma como entrada de esta actividad la documentación del sistema y la lista de servicios descritos en la Historia de Servicio y se define cuáles de estos deben ser probados, según la criticidad de los mismos o la funcionalidad que represente.

En esta guía se recomienda realizar pruebas de todos los servicios que hacen parte del proyecto, pero esto depende del tamaño del mismo, para lo cual se podría basar en algunos criterios como nivel de criticidad o nivel de utilización de los mismos, de esta forma se probarían los que sean más críticos para el proyecto o los que más se utilicen.

3.2.2 Definición de baterías de casos y requisitos de pruebas

En esta fase se definen los casos de prueba necesarios para probar el sistema y se agrupan en una batería de pruebas, según la definición de las pruebas y los requisitos que satisfacen. Se debe definir Suite de Pruebas por cada Servicio y Baterías de Casos por cada operación o método del servicio al cual se le realicen las pruebas. Adicional se puede definir si se realizaran falsos servicios (Mock Service) en casos de no contar con la totalidad de los servicios, ya sea porque la disponibilidad no es permanente o porque el mismo no se encuentra implementado.

Para trabajar con falsos servicios se utiliza la herramienta SoapUI[2], que en base al archivo WSDL puede crear la simulación de las operaciones o solicitudes del servicio, generar una respuesta con un valor especifico y ejecutar un servidor interno al cual se conecta el software al momento de consumir el servicio. En la Figura 3 se observa la ejecución de un falso servicio para un método que realiza la suma de dos números.

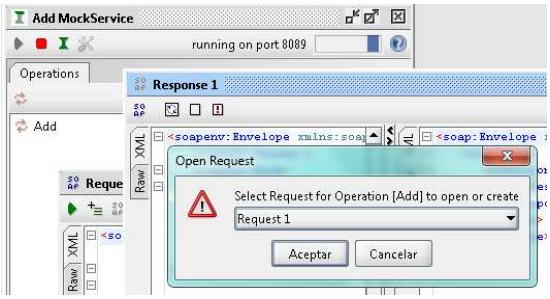

Figura 3. Ejecución del falso servicio en SoapUI
Fuente: Elaboración propia.

A través del proceso anterior se puede garantizar que los desarrolladores puedan consumir el servicio así el mismo no se encuentre desarrollado o disponible.

Durante esta fase se deben definir los resultados de las pruebas que servirán como oráculo[3] de la prueba para la comprobación de las mismas. Estos datos deben ser utilizados en la realización de las baterías de pruebas, al momento de definir las comprobaciones.

3.3 Fase 3 - Implementación de la Prueba de integración de servicios

En esta fase el probador genera los casos de prueba, para lo cual se recomienda en esta guía y por la naturaleza de la prueba, la técnica de grabación e inyección de datos, para lo cual se definen cuatro actividades.

3.3.1 Definición de las plantilla de los casos de prueba

Esta actividad se centra en creación de la estructura del caso de la prueba, que será utilizada para construir el caso de prueba. Esta plantilla se relaciona con la creación automática de pruebas y define la estructura que será utilizada para incluir los datos de la batería de datos al momento de ejecutar la prueba. Para realizar la plantilla se necesitan los requerimientos que cubre el servicio, así como los datos de entrada y salidas del mismo.

Al finalizar esta tarea se obtiene una plantilla del caso de prueba que será utilizada para la construcción del caso de prueba como se observa en la Figura 4. El ejemplo muestra un caso de prueba típico para un servicio, en el cual durante la fase de ejecución deben inyectarse los datos de prueba, reemplazando los signos "?" por valores.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/
   <soapenv:Header/>
   <soapenv:Body>
      <tem:Add>
         <tem:intA>?</tem:intA>
         <tem:intB>?</tem:intB>
      </tem:Add>
   </soapenv:Body>
</soapenv:Envelope>
```

Figura 4. Caso de pruebas en SoapUI
Fuente: Elaboración propia.

La plantilla anterior fue creada utilizando la herramienta SoapUI, que utiliza la dirección de la WSDL de la interface del servicio, para crear el archivo XML.

Igualmente se deben definir las aserciones o comprobaciones del caso de prueba; estas comprobaciones se realizan sobre la respuesta del servicio y pueden ser referentes a la validez del mensaje SOAP, a

[2] SoapUI es una herramienta Open Source pensada para testear el funcionamiento de WebServices. De manera sencilla carga todos los interfaces de los ficheros WSDL o WADL y permite que se puedan lanzar diversos tests.

[3] Oráculo de pruebas se define en (Torres & Escalona, 2009) como un método que provee las salidas esperadas de cada caso de prueba dado; de manera más realista, es una heurística que puede emitir un veredicto de pasa/fallo sobre las salidas de prueba observadas.

la calidad del servicio o a la coincidencia de resultados y utilizan los valores de comprobación definidos en fases anteriores.

3.3.2 La preparación de datos de la prueba

Durante esta tarea se organizan un grupo de datos usados para verificar los requisitos del sistema. Se deben verificar los datos requeridos por el servicio como parámetros de la petición (Request), así como definir los datos que servirán para evaluar el resultado correcto de la respuesta (Response). Estos datos deben ser organizados en una estructura que pueda ser inyectada a las herramientas de carga de datos definida en el ambiente de pruebas. Generalmente se usan archivos CSV, pero la herramienta de carga de datos configurada define el tipo de archivo necesario. La Figura 5 contiene los datos de prueba para un servicio que permite realizar la suma de dos números; los valores se encuentran separados por coma, y contiene el valor de los dos números, y el resultado de la suma.

```
numero1,numero2,suma
2,3,5
3,4,7
10,14,24
18,15,33
13,12,25
```

Figura 5. Archivo de datos de prueba
Fuente: Elaboración propia.

3.3.3 Generación del caso de prueba

El objetivo de esta tarea es crear el caso de prueba utilizando las plantillas de caso y los datos de prueba. Para esta tarea se necesitan los datos de prueba que tendrian las entradas y salidas y la plantilla del caso de prueba generada desde SoapUi como se observa en la Figura 4. En este caso para generar los casos de prueba, se deben ajustar las plantillas para que reciban los datos de prueba inyectados a través de la herramienta de carga de datos PushToTest. La Figura 6 muestra el esquema de los elementos participantes en el proceso de inyección de datos a la plantilla de prueba a través de la herramienta PushToTest.

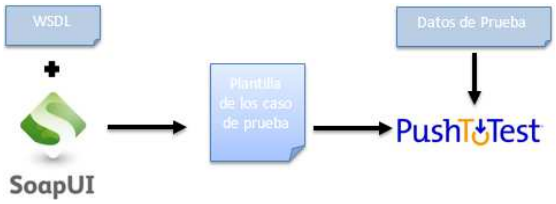

Figura 6. Proceso de inyección de datos
Fuente: Elaboración propia.

3.3.4 Definición del plan de ejecución y la secuencia de comandos

Esta tarea define los elementos necesarios para ejecutar el sistema bajo prueba y realizar la ejecución automática de las pruebas. Se deben iniciar los servidores de ser posible, iniciar los servicios simulados en caso de ser necesario, ya que si los servicios son internos y no se encuentran implementados, se pueden simular para realizar las prueba, igualmente si el servicio es externo y no está disponible todo el tiempo se puede simular para mantener un acceso permanente. Este plan de ejecución se puede definir a través de secuencias de comandos o instrucciones de comandos Ant. La Figura 7 corresponde a una secuencia Ant que realiza el llamado a varios procesos para la ejecución de los distintos elementos que participan en la prueba.

```
<target name="iniciarPruebasSoa" description="Iniciar las Pruebas de SOA">
    <antcall target="iniciar-entorno-pruebas" />
    <antcall target="iniciar-servicios-mock" />
    <antcall target="iniciar-pruebas-soapui" />
    <antcall target="generar-reportes" />
</target>
```

Figura 7. Secuencia de comandos en Ant
Fuente: Elaboración propia.

En esta fase se deben resolver las dependencias para que el ambiente de pruebas se pueda ejecutar de forma independiente a través de la creación de falsos servicios (Mock Services) con herramientas como SoapUI.

3.4 Fase 4 - Ejecución de la Prueba y análisis de resultados

El objetivo de esta fase es verificar el comportamiento del sistema bajo prueba, para lo cual se ejecutan los casos de prueba. Generalmente se realiza a través de la ejecución automática de las secuencias de comandos. Esta fase se divide en tres actividades equivalentes al despliegue del entorno, al despliegue del sistema bajo prueba y a la ejecución de las pruebas.

3.4.1 Despliegue de la plataforma

Esta actividad pone en marcha los elementos requeridos para desplegar el sistema bajo prueba, ya que debido a la naturaleza de las pruebas, estas requieren de la ejecución de otros elementos para proporcionar un contexto de prueba similar al que tendría en el despliegue. Es importante que se desplieguen todos los elementos como Bases de Datos, Servidores de Aplicaciones y demás.

3.4.2 Despliegue del sistema bajo prueba

El objetivo de esta actividad es ejecutar el sistema bajo prueba. Para automatizar esta actividad se ejecutan las secuencias de comandos que permiten dar inicio al sistema que provee los servicios o lo concerniente a los servicios simulados.

3.4.3 Ejercitar el sistema bajo prueba

En esta actividad los datos de prueba se inyectan en el sistema bajo prueba a través del caso de prueba; después las herramientas de prueba ejecutan el caso de prueba y obtienen un conjunto de resultados que expresan el comportamiento del sistema, basados en las condiciones definidas en las validaciones de las plantillas de pruebas.

4 VALIDACIÓN DEL METODO DE PRUEBA DE INTEGRACIÓN PARA ARQUITECTURA ORIENTADAS A SERVICIOS

Con el objetivo de cumplir a cabalidad con el propósito de este trabajo, se realiza la validación del método propuesto por medio de un ejemplo aplicado a un proyecto de desarrollo de software en la Universidad Simón Bolívar sede Cúcuta al cual se le aplicarán cada una de las fases del método diseñado.

El contexto del proyecto es la Universidad Simón Bolívar de la Ciudad de Cúcuta Norte de Santander y con el apoyo del Departamento de Sistemas y su equipo de desarrolladores, quienes posibilitaron la aplicación de este método.

4. CONCLUSIONES.

De la revisión literaria realizada en esta investigación se observa que las pruebas de software tienen gran importancia en la actualidad del desarrollo, que existen diversas técnicas para aplicarlas y que en algunas metodologías estas permiten dar un alto grado de calidad en los aplicativos desarrollados. Referente a las pruebas de integración en arquitecturas orientadas a servicios, las mismas están siendo organizadas y tienden hacia procesos automatizados, integración continua y el uso de herramientas que permitan eliminar en la fase de desarrollo la dependencia generada entre el cliente y el servidor.

Se diseña un método que a través de sus fases guía al equipo de desarrollo para que puedan diseñar e implementar un conjunto de pruebas que se aplican a las interfaces que participan en los proyectos SOA, como parte del proceso de pruebas de integración. El método además de apoyar el proceso de diseño e implementación de las pruebas, intenta documentar el mismo a través del uso de plantillas que generan trabajo

adicional, aspecto que debe evaluarse al momento de conocer el equipo de desarrollo y determinar si el mismo está dispuesto a aplicar el método.

El método propuesto se enfoca en la evaluación de las interfaces que participan en el proyecto apoyándose en un conjunto de herramientas, permitiendo que los distintos desarrolladores puedan trabajar en la capa de entrega al consumir los servicios sin preocuparse por la finalización de la capa de exposición de funcionalidades o servicios.

En la validación del modelo se logró apoyar el proceso de desarrollo de un aplicativo para la Universidad Simón Bolívar sede Cúcuta, pero se presentaron algunos inconvenientes referentes al proceso de configuración de las herramientas para la ejecución de pruebas, debido a que la configuración de las mismas no es tan sencilla, hay poca documentación en español y la persona encargada del proceso de pruebas no había trabajado herramientas como Ant para la generación de Scripts.

Al momento de aplicar las plantillas del proceso de prueba, el equipo encargado del proyecto no estuvo totalmente de acuerdo y se encontró una resistencia evidente, debido a que el equipo encargado de la realización del proyecto está compuesto en su mayoría por desarrolladores y no están acostumbrados a realizar un proceso de pruebas y documentación.

Dentro de los aspectos a destacar se tiene el proceso de desarrollo debido a que con la creación de los falsos servicios, los desarrolladores trabajaron directamente en la aplicación sin preocuparse por la funcionalidad o disponibilidad de los servicios, esto permitió que los procesos tanto a nivel del cliente como a nivel de los servicios se realizaran paralelamente.

4. BIBLIOGRAFÍA

Ahmad K. Shuja, J. K. (2007). IBM Rational Unified Process Reference an Certification Guide: Solution Designer.

Delgado, A., & Garcia, I. (2010). Metodologías de desarrollo para Service Oriented Architectures con Rational Unified Process. Metodologías de desarrollo para SOAs con RUP, 126-131.

Facundo, C. (2015). Automatizacion de Pruebas de Integración en Arquitecturas Orientadas a Servicios. EST 2015. Buenos Aires.

Gelvez, H. A. (2010). Contribucion a la gestion de los procesos de pruebas de software y servicios.

IBM. (2015). IBM Developer Works. Obtenido de http://www.ibm.com/developerworks/ssa/library/ws-SOAbestpractices/

Krafzid, D., Banke, K., & Slama, D. (2005). Enterprise SOA Service-Oriented Architecture: Best Practices. Prentice Hall.

Mark Utting, B. L. (2007). Practical Model-Based Testing a Tools Aproach. San Francisco: Morgan Kaufmann.

Paul Baker, Z. R. (2008). Model-Driven Testing Using the UML Testing Profile. Springer.

SWEBOK. (2004). Guide to the Software Engineering Body of Knowledge.

Torry Harris Bussiness Solutions. (2007). SOA Test Methodology. Obtenido de Torry Harris.