METODO DE PRUEBAS DE INTEGRACIÓN PARA ARQUITECTURAS ORIENTADAS A SERVICIOS

FACULTAD DE INGENIERÍAS Y ARQUITECTURA
UNIVERSIDAD DE PAMPLONA N.S

Pamplona, Enero de 2016

Tesis de Maestría dirigida por la Msc. Nelson Beltrán Galvis y desarrollado por Carlos René Angarita Sanguino para optar el título de Magister en Gestión de Proyectos Informáticos.

AGRADECIMIENTOS

A mi director de tesis, Nelson Beltrán Galvis, por su aporte académico, su conocimiento en el área y por su paciencia.

Al Magister Luis Alberto Esteban Villamizar por su gran colaboración y disposición para que termináramos este proyecto.

A mi esposa Claudia Yamile Gómez Llanez y a mi hijo Carlos René por su apoyo y comprensión.

ABSTRACT

This document describes a method that guides developers in the process of designing and implementing integration tests for service-oriented architectures, ranging from the design phase to the implementation phase.

The method proposed here was validated using an example applied to the case study "WebSimon - System for Information Consultation Students of Simon Bolivar University in the city of Cucuta."

Keywords: Software Test; Service Oriented Architecture; Integration Test.

RESUMEN

Este documento describe un método que guía a los desarrolladores en el proceso de diseño y ejecución de pruebas de integración para arquitecturas orientadas a servicios, que comprende desde la fase de diseño hasta la fase de ejecución de la prueba.

El método aquí propuesto fue validado mediante al caso de estudio "WebSimon – Sistema para la Consulta de Información de Estudiantes de la Universidad Simón Bolívar de la ciudad de Cúcuta".

Palabras Claves: Pruebas de Software; Arquitectura Orientadas a Servicios - SOA; Pruebas de Integración.

1. INTRODUCCIÓN 1
1.1. PLANTEAMIENTO DEL PROBLEMA 1
1.2. ANÁLISIS Y FORMULACIÓN DEL PROBLEMA 1
1.3. JUSTIFICACIÓN 2
1.4. OBJETIVO GENERAL 3
1.5. OBJETIVOS ESPECÍFICOS 3
1.6. OBJETO DE ESTUDIO 3
1.7. ACOTACIONES 3
1.8. DISEÑO METODOLÓGICO 3
1.8.1. Tipo de Investigación, enfoque y método 4
1.8.2. Enfoque Cualitativo 5
1.8.3. Alcance de Estudio 6
1.8.4. Población y Muestra de la Investigación 6
1.8.4.1. Estudio Exploratorio 6
1.8.4.2. Estudio Descriptivo 6
1.8.5. Diseño de la Investigación 6
1.8.5.1. Consecución del primer objetivo: 6
1.8.5.2. Consecución del segundo objetivo: 7
1.8.5.3. Consecución del tercer objetivo: 7
1.8.6. Actividades 7
1.8.7. Fuentes de datos recopiladas 8
1.9. INSTRUMENTOS DE CAPTURA DE INFORMACIÓN 8
2. MARCO TEÓRICO 9
2.1. PRUEBAS DE SOFTWARE 9
2.1.1. Tipos de pruebas 10
2.1.1.1. De acuerdo al ámbito de las Pruebas 10
2.1.1.2. De acuerdo a la dimensión de calidad 11
2.1.2. Ciclos de Aplicación de las Pruebas 11
2.1.3. Procesos de Pruebas 12
2.1.4. Modelado de Pruebas 14
2.2. SERVICIOS WEB Y SOA 15

2.3. PRUEBAS DE INTEGRACIÓN EN SOA 18
2.4. ESTADO DEL ARTE 19
3. MÉTODO DE PRUEBAS DE INTEGRACIÓN PARA ARQUITECTURAS ORIENTADAS A SERVICIOS 23
3.1. DESCRIPCIÓN DE LAS FASES DEL MÉTODO 23
3.1.1. Fase 1 – Conocimiento del entorno de desarrollo 24
3.1.1.1. Configurar el entorno de pruebas para las pruebas de integración de servicios 24
3.1.1.2. Definir el sistema bajo pruebas 24
3.1.1.3. Elaborar plan de pruebas 26
3.1.2. Fase 2: Diseño y especificación de los casos de prueba de integración de los servicios 26
3.1.2.1. Definición de elementos a probar 26
3.1.2.2. Definición de baterías de casos y requisitos de pruebas 27
3.1.3. Fase 3: Implementación de la Prueba de integración de servicios 28
3.1.3.1. Definición de las plantilla de los casos de prueba 28
3.1.3.2. La preparación de datos de la prueba 29
3.1.3.3. Generación del caso de prueba 29
3.1.3.4. Definición del plan de ejecución y la secuencia de comandos 30
3.1.4. Fase 4: Ejecución de la Prueba y análisis de resultados 30
3.1.4.1. Despliegue de la plataforma 31
3.1.4.2. Despliegue del sistema bajo prueba 31
3.1.4.3. Ejercitar el sistema bajo prueba 31
4. VALIDACIÓN DEL MÉTODO PROPUESTO 32
4.1. APLICACIÓN DE LAS FASES DEL MÉTODO 32
4.1.1. Fase 1 – Conocimiento del entorno de desarrollo 32
4.1.1.1. Configurar el entorno de pruebas para las pruebas de integración de servicios 32
4.1.1.2. Definir el sistema bajo pruebas 34
4.1.1.3. Elaborar plan de pruebas 34
4.1.2. Fase 2: Diseño y especificación de los casos de prueba de integración de los servicios 34
4.1.2.1. Definición de elementos a probar 34
4.1.2.2. Definición de baterías de casos y requisitos de pruebas 36
4.1.3. Fase 3: Implementación de la Prueba de integración de servicios 38
4.1.3.1. Definición de las plantilla de los casos de prueba 38

4.1.3.2. Preparación de los datos de prueba 42
4.1.3.3. Generación del caso de prueba 43
4.1.3.4. Definición del plan de ejecución y secuencia de comandos 44
4.1.4. Fase 4: Ejecución de la Prueba y análisis de resultado 45
4.1.4.1. Despliegue de la plataforma 45
4.1.4.2. Despliegue del sistema bajo prueba 45
4.1.4.3. Ejercitar el sistema bajo prueba 46
5. CONCLUSIONES Y RESULTADOS 47
5.1. CONCLUSIONES 47
5.2. RESULTADOS 48
5.2.1. Estado del Arte 48
5.2.2. El método 48
5.2.3. Validación del método 48
5.2.4. Participación en Eventos 49
6. BIBLOGRAFÍA 50
7. ANEXOS 53

LISTADO DE FIGURAS

Figura 1 - Fases del método de pruebas 23

Figura 2 Ejecución del falso servicio en SoapUI 28

Figura 3 Caso de Prueba en SoapUI 29

Figura 4-Proceso de inyección de datos 30

Figura 5 Secuencia de comandos en Ant 30

Figura 6-Aplicación de la plantilla de la historia del servicio 36

Figura 7-Estructura de batería de pruebas en SoapUI 37

Figura 8 Importación de WSDL 38

Figura 9 Plantilla de petición de caso consultar_historico 38

Figura 10-Peticiones de los casos de prueba 39

Figura 11 Plantilla caso de prueba validar_credencial 39

Figura 12 Plantilla caso de prueba verificar_pago 40

Figura 13-Validaciones realizadas sobre casos de prueba 42

Figura 14 Ejemplo de archivo de datos para caso de prueba 42

Figura 15 Paso de valores del archivo de propiedades a la plantilla del caso de prueba 43

Figura 16 Escenario de pruebas en TestMaker 44

Figura 17 Definición de comandos Ant para ejecución de pruebas 45

LISTADO DE TABLAS

Tabla 1-Plantilla de Historia del Servicio 26

Tabla 2-Listado de servicios a probar 27

Tabla 3-Hardware utilizado en el proyecto de pruebas 32

Tabla 4-Ubicación de elementos del software bajo prueba 33

Tabla 5-Tecnología necesaria para la ejecución del software bajo prueba 33

Tabla 6-Estructura de directorios para el almacenamiento de los elementos de las pruebas 33

Tabla 7-Listado de requerimientos del sistema 34

Tabla 8 Listado de servicios a probar 35

Tabla 9 Listado de suite y batería de pruebas 37

Tabla 10 Validaciones a implementar en los casos de prueba 41

1. INTRODUCCIÓN

El producto de este trabajo es un método, instrumento de apoyo para el diseño de pruebas de integración en arquitecturas orientadas a servicios.

Se inicia con un marco teórico basado en diversos autores que definen las pruebas de software como un método para asegurar la calidad en el software a través del uso constante de pruebas, también se indaga sobre la arquitectura orientada a servicios.

Como producto final se obtiene un método que permite a los probadores, gestionar el proceso de diseño y ejecución de las pruebas de integración en aplicaciones orientadas a servicios.

Este documento cuenta con un primer capítulo que define el planteamiento del problema, la justificación y metodología utilizada. El segundo capítulo define el marco teórico de las pruebas de software, la arquitectura orientada a servicios (SOA) y las pruebas en estas arquitecturas. El tercer capítulo define el método de pruebas con las respectivas fases que deben ser aplicadas para el diseño y ejecución de las pruebas de integración. Además el documento cuenta con un capitulo para describir el proceso de validación del método por medio de un ejemplo aplicado a "Web Simón – Sistema para la gestión de información estudiantil en la Universidad Simón Bolívar Sede Cúcuta", ayudando a realizar el proceso de diseño y ejecución de las pruebas de integración de los distintos servicios que componen este proyecto.

1.1. Planteamiento del problema

¿Cómo aplicar pruebas de software en el nivel de integración en aplicaciones con arquitecturas orientadas a servicios?

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; y para ello, 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 (Moreno Ceballos & Morales Oliva, 2008), enfocando sus esfuerzos en asegurar la calidad en el nivel de pruebas de unidades y no en el nivel de pruebas de integración.

1.2. Análisis y formulación del problema

Aunque las pruebas de software están determinadas por la naturaleza de las aplicaciones, es importante saber que las mismas no están claramente definidas debido a la poca madurez de muchos tipos de aplicación, así como las pocas herramientas disponibles. En este proyecto se enfatizó en el nivel de integración de las pruebas de aplicaciones SOA, realizando validaciones

sobre ese nivel intermediario que cubren las interfaces de los servicios y no desde las pruebas de unidad de los elementos que conforman los mismos.

La 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 se deben identificar dos escenario en las Pruebas de Integración a Nivel de Servicios, el primero donde se tiene control de las partes que componen el servicio ya que se puede acceder a los componentes porque estos son de propiedad de la empresa y el segundo cuando los servicios pertenecen a otro equipo por lo tanto solo se puede acceder 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.

1.3. Justificación

A pesar de los avances de las tecnologías de la información y de todos los procesos de documentación de los mismos, cada vez que ingresa una nueva tecnología la aplicación de la misma eleva los riesgos para las empresas, convirtiendo a la etapa de pruebas en parte fundamental del ámbito del proyecto, por tal razón, se hace necesario diseñar un método, que recopile los procesos de la fase de pruebas, que permita asegurar el mejoramiento de la calidad de los productos finales, de los sistema orientados a servicios y reducir al mínimo los errores en la fase de implantación.

La falta de procesos claros y de una cultura de aplicación de pruebas en los proyectos de desarrollo de software y la tendencia al uso de la nuevas tecnologías como los Web Services, han generado inconsistencias y retrasos en los desarrollos de los productos de software, los errores son descubiertos en la etapa de implementación, elevando el costo de mantenimiento y en general el costo total del proyecto; la complejidad inherente que SOA introduce, hace necesaria la participación de arquitectos y la adopción de una metodología.

En los entornos SOA, sistemas y aplicaciones dependen uno del otro para completar un proceso de negocio o servicio. Sin embargo los numerosos puntos de intercambio crean numerosas oportunidades para defectos que deben ser encontrados y corregidos. Las pruebas de integración proveen la visión necesaria para ver lo que está sucediendo dentro del entorno de SOA y validar que los servicios, los mensajes, las interfaces y procesos de negocio se realizan correctamente, es por esto que esta investigación se orientó a la formulación de un método de pruebas de integración con arquitectura orientadas a servicios.

El resultado de esta investigación permitió al personal encargado de la pruebas de software, en proyectos que incluyan el uso de servicios, contar con una herramienta que les delimita el camino a seguir para poder aplicar las pruebas de integración en dichos proyectos, mejorando la calidad de los proyectos con arquitecturas orientadas a servicios a través del uso de herramientas que permitan automatizar las pruebas y definir formalmente las mismas sobre las interfaces de los servicios.

1.4. Objetivo general

Diseñar y proponer un método de gestión de pruebas de Integración para desarrollos de software con Arquitecturas Orientadas a Servicios validado por medio de un ejemplo aplicado al proyecto WebSimon para la gestión de información de estudiantes en la Universidad Simón Bolívar sede Cúcuta.

1.5. Objetivos específicos

Elaborar el estado del arte de procesos, métodos y metodologías de pruebas de software en desarrollos con Arquitecturas Orientadas a Servicios, profundizando en las pruebas de integración.

Formular un método de Pruebas de Integración aplicadas a SOA, basado en metodologías y métodos actuales.

Validar el producto de investigación planteado a través de un ejemplo ejecutando cada uno de los pasos del método a desarrollar.

1.6. Objeto de estudio

El objeto de estudio de este proyecto es la Aplicación de Pruebas de Integración en Aplicaciones Orientadas a Servicios basándose en el área de las Pruebas de Software en el subtema de pruebas basadas en la naturaleza del sistema, enmarcado en el SWEBOK® (Guía para el cuerpo de Conocimiento de la Ingeniería del Software).

1.7. Acotaciones

La validación del producto se hace bajo las condiciones que proporcione la universidad donde se desarrolla el proyecto, al cual se le aplicarán los pasos protocolarios del método.

Las herramientas utilizadas dentro de la investigación serán bajo licencia de software libre.

1.8. Diseño metodológico

El tipo de investigación que se utiliza es cualitativo. Su propósito consiste en "reconstruir" la realidad, tal y como la observan actores de un sistema social previamente definido[1]. Estas investigaciones comúnmente se utilizan para descubrir y refinar preguntas de investigación (Grinell, 1997), en este caso, la relacionada con las pruebas de software en arquitecturas orientadas a servicios. Con frecuencia se basa en métodos de recolección de datos sin medición

1 En esta investigación los actores se relacionan con los desarrolladores de software e ingenieros de pruebas, en un sistema social relacionado a proyectos de desarrollo de software en la Universidad Simón Bolívar.

numérica, para esta investigación las descripciones relacionadas a los procesos de pruebas y metodologías de desarrollo y las observaciones de las labores realizadas para la realización de pruebas en estos proyectos orientados a servicios. Las preguntas surgen como parte del proceso de investigación siendo este flexible, y se mueve entre los eventos y su interpretación, entre las respuestas y el desarrollo de la teoría.

(Silverman, 1995) Citado por (Rodríguez Peñuelas, 2005) hace un análisis comparativo de concepciones y críticas a la metodología cualitativa encontrando una nueva versión de esta metodología, señalando lo siguiente.

1. La preferencia por investigación cualitativa, usa palabras más que números.
2. La preferencia por información que sucede de manera natural y por observación, más que por experimentos y por entrevistas no estructuradas y no por las estructuradas, de cualquier modo esto es relativo.

La razón de este tipo de investigación recoge información de carácter subjetivo, es decir que no se percibe por los sentidos, los valores, aspectos culturales, entre otros. Por lo que sus resultados siempre se traducen en apreciaciones conceptuales. Su diseño no incluye hipótesis, si no formas de entrevistar, observar o grabar videos o lugares o las personas a investigar para luego convertir la información en categorías de análisis, hasta obtener una alta precisión respecto de la realidad investigada.

En base a lo anterior 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 caracterización 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 enseña un ejemplo que permite la validación del método.

1.8.1. Tipo de Investigación, enfoque y método

Para efectos del proyecto, se adopta la propuesta de (Bogdan & Taylor, 1987) por considerarla interesante ya que describen con mayor amplitud la metodología cualitativa, el cual se puede distinguir por las siguientes características: 1.La investigación cualitativa es inductiva. Los investigadores desarrollan conceptos y comprensiones partiendo de pautas de los datos y no recogiendo datos para evaluar modelos, hipótesis o teorías preconcebidos. Los investigadores siguen un diseño de investigación flexible, comenzando sus estudios con interrogantes vagamente formuladas. 2. En la metodología cualitativa el investigador ve al escenario y a las personas en una

perspectiva holística; las personas, los escenarios o los grupos no son reducidos a variables, sino considerados como un todo. Se estudia a las personas en el contexto de su pasado y las situaciones actuales en que se encuentran.

La investigación es de naturaleza cualitativa. En el presente estudio tiene como fin establecer cómo se llevará a cabo la investigación, diseñando detalladamente como las diversas técnicas aplicadas a las pruebas de software pueden ser organizadas en cada una de las fases del método propuesto y así mismo se enseña un ejemplo que permite la validación del método dando respuesta a los objetivos planteados.

A través de la recopilación de documentos y archivos, para contextualizar la situación actual de las pruebas de integración en SOA, así como los principales conceptos de servicios, software y demás. Teniendo presente que existen diversas técnicas utilizadas para aplicar pruebas de software, así como diversos estudios, metodologías y herramientas. Basado en lo anterior se generaran las diversas fases del modelo que permita mejorar la aplicación de las pruebas de integración. A través de un ejemplo se realizara la validación del método elaborado.

1.8.2. Enfoque Cualitativo

Este trabajo se fundamenta en un enfoque metodológico de tipo cualitativo; iniciando con la recolección de datos y fuentes bibliográficas que contienen información referente a las pruebas de software, servicios web, arquitecturas orientadas a servicios y finalmente sobre pruebas de integración en arquitecturas orientadas a servicios que es el objeto de estudio de este proyecto, sin hacer medición numérica para plantear las preguntas de investigación y lograr los resultados esperados; teniendo en cuenta que en los enfoques cualitativos se pueden desarrollar preguntas e hipótesis antes, durante o después de la recolección y el análisis. (Sampieri, Collado, & Lucio., 2010); cabe resaltar que siguiendo el enfoque cualitativo y apoyado del estudio de caso que es una estrategia metodológica de la investigación científica (Carazo, 2006); estrategia que permite en este trabajo a partir de una gran variedad de fuentes construir el cuerpo de la investigación y generar nuevos planteamientos.

Por su parte, la investigación cualitativa da profundidad a los datos, la riqueza interpretativa, la contextualización del ambiente o entorno, los detalles y las experiencias únicas. También aporta un punto de vista "fresco, natural y holístico" de los fenómenos, así como flexibilidad.

Bajo esta perspectiva, el autor, de acuerdo con la experiencia en el estudio de la metodología cualitativa lleva a cabo una gran diversidad de conceptos y características muy particulares aportando herramientas necesarias para llevar a cabo en la investigación, reconocidos por la comunidad científica en el campo de la metodología cualitativa.

Reafirmando lo definido por (Hernández, Fernández, & Baptista, 2010) Hernández, Fernández y Baptista (2010), quienes señalan que el enfoque cualitativo se selecciona cuando se busca comprender la perspectiva de los participantes que para este estudio corresponde a los Ingenieros de pruebas y equipo de desarrollo, acerca de cómo se realizan las pruebas de software en arquitecturas orientadas a servicios.

1.8.3. Alcance de Estudio

Teniendo en cuenta que los estudios cualitativos no pretenden generalizar de manera intrínseca los resultados a poblaciones más amplias; e incluso no buscan que sus estudios puedan replicarse, se fundamentan más en un proceso inductivo (exploran, describen y luego generan perspectivas teóricas) (Gomez, 2006).

El presente estudio se caracteriza por su propósito de identificar las técnicas y métodos para la aplicación de pruebas de software según la naturaleza del mismo, las herramientas y demás elementos necesarios para la aplicación de estas. Comprender desde la base del conocimiento de las pruebas cuales son los elementos que componen todo el ambiente de aplicación de la prueba y brindar al lector una guía para el diseño y aplicación de las pruebas de integración.

1.8.4. Población y Muestra de la Investigación

La muestra cualitativa es la unidad de análisis o conjunto de personas, contextos, eventos o sucesos sobre la cual se recolectan los datos sin que necesariamente sea representativo. Para el presente trabajo se sigue el objeto de estudio de este trabajo y las investigaciones realizadas a través de la estrategia metodológica estudio de caso, el alcance de estudio pueden ser: exploratorio y descriptivo.

1.8.4.1. Estudio Exploratorio

En el presente trabajo se hace investigación exploratoria ya que en el marco de las Pruebas de Software se cuenta con diversos estudios, que definen modelos aplicables; pero muy pocos se encuentran desarrollados y los mismos no se orientan completamente a las pruebas de integración; por lo que se exploran artículos de investigación, trabajos y demás aportes bibliográficos con el fin de dar nuevas expectativas al diseño del método propuesto.

1.8.4.2. Estudio Descriptivo

En este trabajo se hace investigación descriptiva desde que se inicia con la caracterización del proceso de aplicación de pruebas de software, así como de todo el marco de las pruebas y los elementos que participan en el mismo.

1.8.5. Diseño de la Investigación

Para el presente trabajo el enfoque de estudio es de tipo cualitativo, por lo cual el método propuesto se caracteriza por ser abierto, flexible y construido durante la realización del estudio; basado en teorías, aportes y experiencias de otros investigadores del área de las pruebas de software.

Así mismo es necesario describir la consecución de los objetivos específicos del presente estudio, teniendo en cuenta el proceso inductivo (exploran, describen y luego generan perspectivas teóricas) (Gomez, 2006).

1.8.5.1. Consecución del primer objetivo:

Para elaborar el estado del arte de las pruebas de software, se utilizará el método histórico, iniciando con una búsqueda exploratoria de artículos de investigación, autores, trabajos de grado y documentos que aportan al cuerpo del conocimiento de la Ingeniería del Software y cuyo contenido específico sea técnicas de pruebas de software, permitiendo en orden cronológico la

clasificación en tres etapas: pasado, presente y futuro, con el fin de conocer el avance que ha tenido el objeto de estudio en el transcurso del tiempo.

La construcción del estado del arte se basa en la exploración de documentos que es una metodología denominada Revisión literaria; que consiste en la recopilación de documentos y archivos, para contextualizar la situación actual de las técnicas de pruebas de software y así aportar nuevas teorías al objeto de estudio.

1.8.5.2. Consecución del segundo objetivo:

Para definir un método que haga parte del cuerpo del conocimiento de la ingeniería del software y de las pruebas de integración de software en SOA; se parte de la fundamentación del capítulo de marco teórico y estado del arte; basado en la revisión exhaustiva de conceptos, metodologías propuestas por algunos investigadores y metodologías aplicadas en algunos estudios de caracterización. Aplicando la metodología revisión literaria, se observa que no hay un método que permita diseñar y aplicar las pruebas de software en estas arquitecturas orientadas a servicios, solo se observan diversas herramientas, algunas técnicas y procesos aislados.

La investigación cualitativa se basa, en el proceso mismo de recolección y análisis. Y recordando que ante todo es interpretativa, ya que el investigador hace su propia descripción y valoración de los datos (Sampieri, Collado, & Lucio., 2010). Para el logro de este objetivo se realizará un análisis de las distintas técnicas y métodos que hay para la aplicación de las pruebas y como resultado se propone un conjunto de pasos que incluyen la creación de formatos, el uso de herramientas y técnicas que permitan organizar, mejorar y automatizar el proceso de la aplicación de las pruebas en esta naturaleza de aplicaciones. Este método diseñado con las pautas necesarias para aplicarse a proyectos de software SOA pretende mejorar la calidad del software.

1.8.5.3. Consecución del tercer objetivo:

La validación del método diseñado; se hace mediante un ejemplo aplicado a un proyecto de desarrollo de software en la Universidad Simón Bolívar de Cúcuta; al cual se le aplicarán cada una de las fases del método diseñado, buscando la validez del método creado a través de la estrategia de investigación **estudio de caso**, que permita dar credibilidad a la investigación y verificar el funcionamiento del método. El contexto en el cuál se desarrollará el ejemplo de validación del método son las instalaciones de la Universidad Simón Bolívar de la Ciudad de Cúcuta Norte de Santander, bajo el apoyo del Equipo de Sistemas de la Universidad.

1.8.6. Actividades

1. Revisión bibliográfica de las características y funcionamiento de las Arquitecturas Orientadas a Servicios, sus metodologías, procedimientos e implementación.
2. Indagar sobre procesos, métodos y metodologías de pruebas de software en desarrollos con Arquitecturas Orientadas a Servicios, profundizando en las pruebas de integración.
3. Análisis de las técnicas de pruebas de software teniendo en cuenta la naturaleza de desarrollo de software y el ámbito de las mismas.
4. Elaboración del estado del arte de las técnicas, herramientas y métodos de las pruebas de software teniendo en cuenta la evolución.
5. Diseñar el método para la gestión de la pruebas de integración en SOA, con cada una de sus fases y actividades necesarias para su implementación.

6. Seleccionar el proyecto de desarrollo de software al cual se le aplicarán cada una de las fases propuestas en el método a diseñar.
7. Comprobar la funcionalidad de la propuesta mediante un ejemplo aplicado a un proyecto de la Universidad Simón Bolívar sede Cúcuta mediante la aplicación de cada una de las fases diseñadas en el método propuesto.
8. Revisar los resultados obtenidos de la aplicación del ejemplo al método propuesto y generar conclusiones y resultados productos de la interpretación, observación y de análisis descriptivos entre otros métodos empíricos.

1.8.7. Fuentes de datos recopiladas

Para la extracción y recopilación de datos fuentes, se toma en cuenta las referencias pertinentes para el problema de investigación, iniciaciando con el marco teórico, el cual se basa en la integración de la información recopilada.

Siguiendo las buenas prácticas se usa el Método del Mapeo el cúal consiste en ordenar la información recopilada de acuerdo con uno o varios criterios lógicos y adecuados al tema de la investigación. Para este trabajo de investigación se utilizarón documentos en archivos y carpetas para recopilar la información, se ordenaron de acuerdo a cada capítulo del trabajo de investigación.

1.9. Instrumentos de captura de información

En este proceso se realizó la recolección de información basado en la revisión literaria de autores reconocidos en el marco del área de investigación.

2. MARCO TEÓRICO

En esta sección el proyecto de investigación se enmarca en las bases conceptuales y teóricas de las Pruebas de Software, Servicios Web y Arquitecturas Orientadas a Servicios, los procesos y métodos de pruebas de software, los aportes y proyectos relacionados con este trabajo de investigación y el estado del arte de las pruebas de software.

2.1. Pruebas de Software

Para entrar en el contexto de las Pruebas de Software se debe responder la pregunta **¿Qué es una Prueba de Software?;** y para ello se cuenta con un gran número de definiciones entre las más destacadas se tiene la que proporciona la guía SWEBOK®, 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 objetivo de evaluar y mejorar la calidad del producto, identificando defectos y problemas.

Las Pruebas de Software se han convertido en una actividad de gran importancia dentro de la Ingeniería de Software, elemento que brinda la oportunidad detectar fallos y evaluar las características del software. Las pruebas regulan la ejecución de los proyectos y garantizan la calidad del software desarrollado.

En (Estrada Martinez, 2010) se precisa que las pruebas de Software han tenido un crecimiento incremental y una evolución constante, desde el modelo de desarrollo en cascada hasta la aparición de las metodología agiles, las pruebas 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. Inicialmente la prueba fue confundida con depuración[2] y se llevaba a cabo para tomar confianza de que el sistema se ejecutaba correctamente, al poco tiempo paso a la demostración, donde se verificaba que hiciera las cosas bien, y antes de terminar los 80, guiados por G. Myers, se comenzó el periodo orientado a la destrucción, donde el objetivo era encontrar errores, continuando con el periodo orientado a la evaluación y finalizando con el de la prevención.

(Serna & Arango, 2011) Trata sobre el incremento del desarrollo de productos de software, y las tendencias de aseguramiento de la calidad en los mismos, además se considera 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.

---

2 Depuración es el proceso de identificar y corregir errores de programación. En ingles se conoce como Debugging - https://es.wikipedia.org/wiki/Depuraci%C3%B3n_de_programas

Un caso concreto son los métodos XP[3], Scrum[4] y OpenUP5 que aplican ciclos de desarrollo iterativos y que se basan en los principios definidos por el modelo de desarrollo guiado por pruebas (TDD[6]).

2.1.1. Tipos de pruebas

Las pruebas son una disciplina compleja, conformada por muchos elementos técnicos y de gestión, los cuales permiten clasificar las pruebas desde distintos puntos de vista (Gelvez, 2010):

2.1.1.1. De acuerdo al ámbito de las Pruebas

De acuerdo al ámbito de las pruebas o etapas de las pruebas (Paul Baker, 2008) se dividen:

- **Pruebas unitarias.** Su objetivo es probar las unidades más pequeñas del software, el componente o módulo de software. Centran su actividad en verificar la funcionalidad y la estructura (lógica interna) de cada elemento individualmente, una vez que ha sido codificado. Se realizan en el entorno del desarrollador.
- **Pruebas de componentes.** Son un tipo de pruebas unitarias, realizadas por los desarrolladores para probar que el código no sólo compila con éxito, sino que la funcionalidad básica de los componentes y las funciones dentro de un servicio son conformes con las especificaciones. El objetivo de la prueba del componente es coger la pieza más pequeña del software a probar, aislarlo del resto del código, y determinar si se comporta exactamente como se espera. Cada componente se prueba por separado antes de integrarlo en un servicio o servicios. Adicionalmente se hacen revisiones formales del código para asegurar la conformidad con estándares de la organización e identificar debilidades potenciales acerca a la seguridad.
- **Pruebas de integración.** Comprenden verificaciones asociadas a grupos de componentes, generalmente reflejados en la especificación de diseño del sistema y/o subsistemas, y culminan probando el sistema como conjunto. Se centran en verificar el ensamblaje correcto entre los distintos componentes, una vez que han sido probados unitariamente. Se efectúan en el entorno del desarrollador.
- **Pruebas de sistema.** Son pruebas de integración del sistema completo. Permiten probar el sistema en su conjunto y con otros sistemas con los que se relaciona para verificar que las

---

3 XP es una metodología de desarrollo ágil que pone mayor atención en l adaptabilidad que en la previsibilidad - https://es.wikipedia.org/wiki/Programaci%C3%B3n_extrema

4 SCRUM es una metodología desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos. Está especialmente indicada para proyectos con un rápido cambio de requisitos. - http://www.cyta.com.ar/ta0502/v5n2a1.htm

5 OpenUP es una metodología que define el proceso mínimo y suficiente, lo que significa que solo el contenido fundamental y necesario es incluido. - https://es.wikipedia.org/wiki/OpenUP

6 TDD - Test Driven Development es una técnica de diseño e implementación de software incluida dentro de la metodología XP - http://librosweb.es/libro/tdd/capitulo_2.html

especificaciones funcionales y técnicas se cumplen. Se realizan en un entorno fuera del alcance del desarrollador (Johanna Rojas Rojas, 2007) y pueden ser de Rendimiento, Resistencia, Robustez, Seguridad, Usabilidad e Instalación.

- **Pruebas de validación.** Son las pruebas realizadas sobre un software completamente integrado para evaluar el cumplimiento con los requisitos especificados.
- **Pruebas de aceptación.** Los usuarios prueban el sistema o aplicación para establecer si está listo para desplegarlo.

2.1.1.2. De acuerdo a la dimensión de calidad

De acuerdo con las dimensiones de calidad que se desean evaluar, en (Ahmad K. Shuja, 2007) las pruebas se clasifican como:

- Las pruebas funcionales, que a su vez se dividen en pruebas de requisitos funcionales, pruebas de control de acceso y seguridad, y pruebas de volumen.
- Las pruebas de usabilidad evalúan la facilidad de uso de las interfaces con el usuario.
- Las pruebas de confiabilidad involucra la ejecución de otros tipos de pruebas.
- Las pruebas de integridad, pruebas estructurales y pruebas de estrés.
- Las pruebas de rendimiento incluyen: pruebas de contienda, pruebas de carga y pruebas de perfiles.
- Las pruebas de infraestructura se dividen en pruebas de configuración y pruebas de instalación
- Las pruebas de regresión intentan descubrir errores después de un cambio.

2.1.2. Ciclos de Aplicación de las Pruebas

Para resolver la complejidad de la ejecución de las pruebas de software, estas se tratan como un proyecto relacionado con el proceso de desarrollo. En este sentido se divide su aplicación en fases, conformando un ciclo de pruebas, que en (Paul Baker, 2008) se definen brevemente como:

- Requisitos de las pruebas. Se definen todos los recursos necesarios para la ejecución de las pruebas como: herramientas, roles, tiempo, elementos del entorno (requisitos, especificaciones funcionales, modelos y códigos del sistema a probar).
- Diseño de las pruebas. En esta fase se identifican los siguientes aspectos: ítem de prueba, el enfoque de prueba detallado y los casos de prueba de alto nivel asociados. Se seleccionan y derivan los casos de prueba. El resultado es un documento que contiene el diseño de las pruebas, específicamente definiendo la estructura de la suite de pruebas y la relación con el sistema bajo pruebas.
- Especificación de las pruebas. Adiciona datos concretos al diseño de pruebas, sobre los casos de pruebas y las especificaciones del procedimiento de pruebas. La finalidad es detallar las condiciones de pruebas a las especificaciones del diseño y como ejecutar cada

prueba incluyendo por ejemplo los procedimientos de inicio y finalización, así como los pasos de prueba que deben seguirse.

- Implementación de las pruebas. Comprende la generación de las pruebas a partir de las especificaciones, del modelo, o del sistema mismo. Como resultado se obtendrán los artefactos del sistema bajo prueba tales como la configuración de las pruebas y el contexto de las pruebas.
- Validación de las pruebas. Esta actividad verifica la conformidad de las pruebas. Esto implica verificar la conformidad interna tanto con las especificaciones del sistema como con el modelo del sistema. La conformidad interna, consiste en verificar que estén definidos todos los casos de pruebas (suficientes), así como los datos de prueba necesarios, que los procedimientos de prueba no presenten bloqueos, que la sintaxis y la semántica de las pruebas indicadas esté conformes de acuerdo a las reglas del Sistema.
- Ejecución de las pruebas. Comprende todos los pasos necesarios para ejecutar de forma manual o automática las pruebas. La ejecución manual debe estar apoyada por herramientas guiadas por procedimientos de pruebas. La ejecución automática necesita la generación de scripts de pruebas junto con los ejecutores de pruebas. Adicionalmente se usa una plataforma de pruebas para ejecutar y registrar el seguimiento automáticamente. Tanto las pruebas manuales como las automáticas se analizan subsecuentemente.
- Evaluación de las pruebas. Esta actividad implica la comparación de los resultados esperados versus los obtenidos del Sistema bajo prueba, durante la ejecución de las pruebas. Estas respuestas incluyen salidas gráficas, cambios de datos y de estados internos, informes generados y comunicación con el entorno. Usualmente la comparación se hace por un oráculo que se define en el sistema por el árbitro y las acciones de validación. Como resultado de este análisis obtenemos el conjunto de veredictos.

La propuesta anterior de definición de las actividades de ciclo de pruebas, describe muy brevemente estas en función de las herramientas que pueden soportarla, orientadas hacia la generación automática de pruebas a partir de modelos. Sin embargo no detalla las actividades que forman parte de cada fase, ni como fluye la información entre ella, por lo tanto es difícil de aplicar a cualquier dominio. Al respecto en (Gelvez, 2010) se proponen varios enfoques de ciclo de vida de prueba; estos trabajos describen el proceso de prueba como un grupo de cuatro etapas: planificación, diseño, ejecución y revisión de los resultados de prueba. Adicionalmente definen cuatro artefactos principales: el plan, el documento de diseño, informe de ejecución y el informe final de pruebas. Estos trabajos intentan proponer un ciclo de vida de pruebas más general, sin embargo su descripción aún es de alto nivel; y para su implementación deben detallarse las actividades que los implementan.

2.1.3. Procesos de Pruebas

Respecto de la aplicación del ciclo de pruebas, existen muchos procesos que prescriben la manera de generar y aplicar las pruebas. En (Mark Utting, 2007) se describen los más usados en la industria, entre los cuales mencionamos: manual, de grabación, basado en scripts, automatizado, dirigido por datos y basado en modelos.

- Proceso de pruebas manual. Es el estilo más antiguo para probar una aplicación, y aún se usa ampliamente. Tanto el diseño del caso de prueba como su ejecución se hacen de manera manual, y los detalles de bajo nivel respecto de la interacción con el sistema bajo prueba se dejan a cargo del probador-ejecutor. Es sencillo y económico de iniciar, pero a medida que el código evoluciona es difícil realizarle seguimiento.

- Proceso de pruebas de grabación. Consiste en grabar la interacción con el software bajo prueba (Software Under Test - SUT[7]), para luego usarla como una plantilla que se puede ejecutar varias veces con diferentes valores, o adaptarla según el objetivo de la prueba. La grabación se hace por medio de herramientas especializadas. Es flexible en la aplicación, la primera aplicación es costosa, permite pruebas automáticas de regresión, no son muy exigentes y se dificulta medir la cobertura.

- Proceso de pruebas basado en secuencias de comandos. Se basa en un archivo que contiene un conjunto de comandos que ejecutan uno o más casos de prueba. Requiere programar la prueba, ejecuta la prueba de manera automática, facilita las pruebas de regresión[8] automáticas.

- Proceso automatizado de pruebas dirigido por datos claves. Consiste en escribir los archivos de comandos de manera genérica y luego durante la ejecución usar los datos de prueba para particularizar la prueba. Para ello el caso de prueba se define de lo más abstracto posible y luego adaptarlo a los objetivos de la prueba durante su ejecución. La ejecución automática permite aplicar pruebas de regresión.

- Proceso de pruebas basado en modelos. En este proceso se usan herramientas especializadas para generar a partir de los modelos arquitectónicos del sistema bajo prueba los casos de prueba, como en (Gonzalez, 2009) se cita el método para generar casos de prueba tomando como base casos de uso enunciado por (Heumann), e identificando dentro de cada uno los posibles escenarios, o caminos de ejecución, y por último definir los valores a probar de cada caso de prueba. O la propuesta de (Riebisch, 2003) que está centrada en la transformación automática de un modelo de casos de uso a un modelo de uso que sirve como entrada para realizar pruebas estadísticas automáticas. Así el diseño de las pruebas es automático, permite la ejecución automática de pruebas de regresión, cobertura sistemática, medidas de cobertura del modelo y de los requisitos; algunas desventajas se relacionan con la sobrecarga de modelos, no hay suficientes herramientas, no se dispone de modelos para todo el código. El proceso de pruebas basado en modelos se presenta, como una solución que satisface estas necesidades. Sin embargo no es aplicable en todos los casos debido a varias circunstancias, entre otras a la carencia de un modelo del sistema a probar, dificultad (costos) para tener acceso a las herramientas y la metodología de desarrollo aplicada.

---

7 SUT – Software Under Test – corresponde a un término del UML Testing Profile para definir el sistema o software bajo prueba.

8 Las pruebas de regresión intentan descubrir errores después de un cambio.

2.1.4. Modelado de Pruebas

Existen varios lenguajes de modelado que pueden aplicarse para modelar procesos de diferentes dominios de aplicación. Sin embargo dada la importancia que tienen las pruebas de software, a partir de la especificación UML se derivó un estándar de modelado específico para modelar los elementos del entorno de las pruebas; es conocido como *UML Testing Profile* (UTP) (OMG, UML Testing Profile, 2010).

En la especificación se parte de la definición de una arquitectura de pruebas. Que es un conjunto de conceptos que describen los aspectos estructurales de un contexto de pruebas, y que cubre los siguientes componentes:

- Contexto de prueba. Un conjunto de casos de prueba junto a una configuración de la prueba, que conforma la base sobre la que se ejecutan las pruebas.
- Configuración de la prueba. Un conjunto de objetos componentes de prueba y de las conexiones de estos componentes con el sistema bajo prueba.
- Prueba de componente. Es una clase de un sistema de prueba. Contiene un conjunto de interfaces por las que se comunica con otros componentes o con el sistema bajo prueba.
- Sistema bajo prueba. Es el sistema, o componente que está siendo probado.
- Arbitro. Es el propietario del caso de prueba quien evalúa el resultado de la prueba.
- Programación. La programación se usa para controlar la ejecución de los diferentes componentes de pruebas.
- Partes útiles. Representan un conjunto de componentes que ayudan a los componentes de prueba a realizar sus pruebas de comportamiento.

Comportamiento de la prueba. Lo constituyen el conjunto de conceptos para describir el comportamiento de la prueba, sus objetivos, y la evaluación del software bajo prueba; que en la especificación se definen así:

- Control de la prueba. Es una descripción de cómo el sistema bajo prueba debe probarse con el contexto de prueba dado.
- Caso de prueba. Especifica la prueba del sistema, incluyendo la entrada, el resultado y las condiciones de la prueba. En otras palabras implementa un objetivo de la prueba.
- Invocación de la prueba. Se refiere a los parámetros específicos con los que se invoca la prueba.
- Objetivo de la prueba. Describe el elemento que será probado y se asocia a un caso de prueba
- Estimulo. Son datos de prueba que se envían al software bajo prueba para evaluar su reacción.
- Observación. Son los resultados de un estímulo enviados al software bajo prueba.

- Coordinación. Mecanismos para controlar el orden en que se ejecutan las pruebas.
- Defecto. Es el comportamiento generado por la observación de la prueba y que no es manejado por el comportamiento de la prueba en sí misma.
- Veredicto. Es la evaluación de la exactitud del software bajo prueba.
- Acción de validación. Es una acción para evaluar el estado de la ejecución de un caso de prueba.
- Acción de registro. Es una acción para registrar información en el registro de pruebas.
- Registro de prueba. Es una interacción como resultado de la ejecución de una prueba. Representa los mensajes intercambiados entre los componentes de la prueba y el sistema bajo prueba y los estados de los componentes involucrados.

Otro aspecto importante son los datos de prueba, que agrupa un conjunto de conceptos que describen los datos usados en las diferentes interacciones con el sistema bajo prueba.

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 desacopládamente 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[9] y WSDL[10]) 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

9 SOAP Simple Object Access Protocol es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML

10 WSDL Web Services Description Language es un formato XML que se utiliza para describir servicios Web.

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).

En (Gelvez, 2010) se definen los elementos más importantes del modelo SOA:

- **SOA** es un modelo de diseño, que se basa en la encapsulación de la lógica de la aplicación dentro de los servicios que interactúan por medio de un protocolo común; una aplicación específica de SOA son los servicios Web. SOA introduce una capa lógica en la arquitectura, que a través de una interface estándar, proporcionada por los servicios Web establece un punto común de integración. Por lo tanto los servicios Web conforman una pieza esencial en el desarrollo SOA. Esto implica que para desarrollar un SOA, se deben aplicar los principios de diseño de servicios Web y la tecnología que los implementa.
- **Servicio.** 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).
- **Servicio Web.** Son servicios que tiene dos características fundamentales, primero que se comunican por Internet a través de protocolos (HTTP) y segundo que envía y recibe datos en formato XML. Su popularidad en la industria ha llevado al establecimiento de tecnologías estándares. Básicamente se espera que estas tecnologías proporcionen una descripción del servicio y que le permitan transportar documentos XML sobre el protocolo SOAP. Adicionalmente un servicio Web puede actuar como consumidor o proveedor de un servicio, y debe ser localizable para lo cual es necesario registrarlo.
- **Descripción de servicios.** Uno de los aspectos fundamentales de los servicios es que puedan usarse por otros servicios o programas. Un servicio que desea usar a otro debe tener conocimiento de su existencia. Esto se logra a través de las descripciones de los servicios. La descripción más elemental del servicio establece el nombre del servicio y los datos que recibe y que entrega el servicio. Los servicios se describen por medio de WSDL ("Web Service Definition Language") de dos maneras, abstracta y concreta. La descripción abstracta establece las características de la interface del servicio Web sin referirse a los

detalles técnicos. La descripción abstracta se conforma por tres partes "portType", la operación y el mensaje. La descripción concreta permite ejecutar la lógica del servicio. Esta descripción vincula la interface con un protocolo de transporte físico. La descripción concreta se conforma de tres partes enlace, puerto y servicio.

- **Descubrimiento de servicios.** Este es un mecanismo necesario en las aplicaciones SOA, debido a que estas se componen de múltiples servicios que interactúan entre sí y la cantidad de servicios se incrementa dentro y fuera de la empresa. El mecanismo se basa en crear y mantener un directorio o registro de las descripciones de los servicios, al que pueden acceder tanto personas como aplicaciones. Este registro puede ser público o privado. El público permite registrar servicios de diferentes organizaciones que dispongan servicios para ofrecer; y el privado generalmente se implanta dentro de las empresas y proporciona un repositorio para todos los servicios que produce, alquila o compra la empresa.

- **Composición de servicios.** La lógica de los procesos de negocio puede expresarse a través de la definición de un orden lógico en que se deben invocar los servicios. Los procesos de negocios se componen de un conjunto coordinado de llamados de servicios y actividades relacionadas para producir un resultado. Para componer un proceso de negocio, SOA se basa en diferentes tecnologías, como BPEL[11] que ofrece flexibilidad y fácil adaptación a los cambios que se introducen en los procesos. Un proceso BPEL expone su funcionalidad de la misma manera que un servicio Web, a su vez este servicio podría formar parte de otro proceso más complejo. Con lo cual cada proceso BPEL debe ser descrito por un documento WSDL.

En (Paul Baker, 2008) se enuncian los tres estándares más usados con frecuencia para el soporte de Servicios Web:

- **WSDL** (Web Services Definition Language) – una definición de lenguajes basada en XML para los servicios web. En (Gelvez, 2010) se define como el mecanismo utilizado por los servicios web para exponer su funcionalidad por medio de un documento en el cual se define toda la información relativa a este. La industria ha adoptado el estándar WSDL que se ha convertido en el mecanismo más utilizado para describir servicios Web.

  Un documento WSDL utiliza XML para conseguir una gramática estructurada, extensible y fácil de leer parar describir los servicios. Por otra parte, gracias a la utilización de XSD se consigue concretar los tipos de datos intercambiados entre consumidor y proveedor sin depender de una tecnología específica de manera que se conozca el modo exacto de interactuar en términos de intercambio de mensajes.

- **SOAP** (Simple Object Access Protocol) – un mensaje basado en formato XML que puede integrarse con una variedad de mecanismos de transporte. Establece un formato estándar

11 Business Process Execution Language (BPEL) es una forma de orquestar procesos de negocio basada en estándares, compuesto por servicios. http://blog.continuum.cl/bpel-o-esb/

para los mensajes, que básicamente consiste en un documento XML, capaz de contener una llamada al procedimiento remoto (RPC), y estructuras de datos. Esto facilita el intercambio síncrono y asíncrono de datos. Con WSDL establece un formato estándar para la descripción de interfaces para las aplicaciones.
Los mensajes utilizados por SOAP se componen de dos partes; la cabeza y el cuerpo. En la primera encontramos metadatos con información sobre enrutamiento, transacciones u otra información sin limitaciones de ningún tipo.
En la segunda, también conocida como carga útil del mensaje SOAP se encuentra un mensaje XML estándar que contiene la información a trasmitir. Por último, todo el mensaje será trasmitido a su vez como carga de otro protocolo superior que normalmente es HTTP.

- **BPEL4WS** (Bussiness Process Execution Language for Web Services) – un lenguaje de orquestación de flujos de trabajo para definir flujos entre servicios que realizan un proceso de negocio. Es un lenguaje basado en XML para la estandarización de los procesos de negocio en un entorno distribuido que permite a diferentes empresas o áreas de negocio conectar sus aplicaciones y compartir datos. BPEL permite definir detalladamente los procesos del negocio; puede definirse como un lenguaje de programación, pero usualmente se genera de forma automática desde diagramas de flujo. Los comandos de BPEL se llaman actividades.

2.3. Pruebas de Integración en SOA

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.

2.4. Estado del arte

La disciplina de las pruebas de software ha alcanzado un nivel de madurez y esto se ve reflejado en un marco teórico compuesto por estándares, técnicas, buenas prácticas y herramientas que facilitan su aplicación. Sin embargo en los nuevos enfoques de desarrollo de software puede existir la necesidad de actualizar el actual planteamiento. En esta sección se presentan algunos trabajos que tratan de Web Services y la aplicación de pruebas de software, esencialmente las relacionadas con aplicaciones SOA.

En (Hernandez, 2002) se realiza una caracterización de las técnicas de pruebas de software que se basa en dos problemas para los desarrolladores, la primera es el desconocimiento de la cantidad de técnicas existentes y la segunda de los resultados obtenidos con la aplicación de dichas técnicas. En este trabajo propone ser una ayuda para los desarrolladores en la escogencia de la técnica de prueba a utilizar. Se describen sistemáticamente todas las técnicas de pruebas centrándose fundamentalmente en aspectos pragmáticos de las técnicas, lo que permite selecciones más objetivas.

En (Dustdar, 2004) se describe un sistema de prueba para SOA, así como un metalenguaje en XML, que se utiliza para describir los casos de prueba para el sistema de pruebas. El objetivo es dar el implementador una herramienta de gran alcance, por lo que la prueba se puede realizar en primeras etapas del ciclo de desarrollo. El concepto se diseñó de una manera que el sistema de prueba puede también ser utilizado para monitorear el sistema una vez que sea desplegado.

Un trabajo importante sobre la generación de casos de prueba funcional de servicios se expone en (Sinha A., 2006). En él se define un procedimiento para extraer la información necesaria de la especificación de las operaciones de comportamiento (WSDL-S) y con ellas generar un conjunto de casos de prueba. Este trabajo se centra en la generación de casos de prueba; sin embargo existen otras fases que deben cubrirse como parte de un proceso de pruebas.

En la definición de una metodología de pruebas debe tenerse en cuenta que existen diferentes vistas en cuanto al interés de los participantes. De acuerdo con esta premisa en (Canfora G., 2006) se indican las perspectivas de acuerdo con los roles que participan en el modelo SOA y que técnicas se deben aplicar para satisfacerlos, de la misma forma se describen los niveles de aplicación de pruebas SOA. Al respecto en (Ribarov L., 2007) se describen las dificultades de las pruebas a nivel unitario, de integración y funcional en las aplicaciones SOA. Igualmente se dan algunas indicaciones de cómo solucionarlas teniendo en cuenta la complejidad del modelo. Esto es útil para la construcción de un modelo metodológico de pruebas.

En (Laguado, 2007) se presenta una vista general del estado del arte de los servicios web, definiendo su arquitectura, bajo el soporte bibliográfico web service architecture publicada por la W3C (World Wide Web Consortium), se realiza una descripción del protocolo de mensajería de los servicios, de los estándares para la descripción de sus operaciones y su publicación, la codificación para su transporte sobre http, y finalmente se describen algunos de los mecanismos de seguridad utilizados por esta tecnología.

En (Torry Harris Bussiness Solutions, 2007) se realiza una revisión de las diversas pruebas aplicadas a los desarrollos de arquitectura SOA, así como los diversos productos empresariales y libres que se utilizan para la realización de este tipo de pruebas.

Respecto de la definición de modelos de prueba en (Lenz Ch., 2007)se propone un modelo específico de pruebas SOA dirigidas por modelos. En este trabajo se definen las etapas de diseño de pruebas, generación de casos de prueba y ejecución; para las que se definen las actividades necesarias para aplicar las pruebas. Esta aproximación a pesar de describir una técnica específica, propone unas fases que pueden tomarse como punto de partida para definir un proceso más general en cuanto a la aplicación de técnicas.

En (Jing G., 2008) se propone un modelo y un proceso de pruebas de componentes de software, para ser ejecutado por terceros, quienes reciben como entrada el componente a probar y los metadatos que expresan su funcionamiento junto con los requisitos que satisfacen. Como resultado se generan todas las actividades para probar el software (planificación, diseño, elaboración y ejecución del caso de prueba). Esta propuesta muestra un escenario que puede aplicarse para probar aplicaciones SOA, debido a la forma como se desarrollan (distribuidas y a que se usan servicios desarrollados por terceros). Sin embargo bajo el modelo propuesto, el control de los artefactos generados durante las pruebas y los procedimientos aplicados queda fuera del alcance del responsable del desarrollo de la aplicación.

En (Londesbrough, 2008) se proponen de manera genérica las características a tener en cuenta en la definición de un proceso de pruebas para adaptarlo a diferentes enfoques de desarrollo; como por ejemplo SOA. Allí se indican los elementos básicos que deberían tenerse en cuenta en la definición de un proceso, se define el concepto general de un proceso de pruebas, se sugieren las entradas, las salidas y las actividades concretas que deberían detallarse en una propuesta de proceso de pruebas, igualmente se caracterizan todos los elementos de proceso de un entorno de pruebas.

La construcción de aplicaciones SOA se basa en gran parte en la aplicación de las técnicas usadas en el desarrollo de Servicios Web. En consecuencia las técnicas de pruebas de Servicios Web y las herramientas también se aplican en este entorno. En (Yoon H., 2008) se propone un procedimiento paso a paso para probar las conexiones que se establecen durante la orquestación de servicios bajo SOA y se define un procedimiento basado en el soporte que da la herramienta SoapUI. Este tipo de soluciones es fundamental cuando se aplica como parte de un proceso que soporta una metodología. Es en este sentido en el que dentro del modelo metodológico propuesto, se pueden incluir procedimientos similares para ampliar el alcance de las pruebas a aplicaciones SOA.

En (Tsai & Jerry, 2008) se proponen varios criterios de evaluación de la capacidad de prueba de software SOA , que sirve de referencia tanto para los proveedores de servicios y los creadores de aplicaciones para evaluar la prueba apoyo al software SOA .

En (Dueñas & Garcia, 2009) se propone un modelo metodológico general de cómo aplicar las pruebas a las aplicaciones desarrolladas bajo SOA. Este modelo permite la definición de los procesos y de los artefactos y facilita la ejecución de las pruebas a las aplicaciones SOA.

En (Gelvez, 2010) se propone un marco metodológico para la gestión de procesos y proyectos de pruebas de software y servicios basado en el modelado de procesos y en las herramientas que los implementan, realizando una división de aéreas de las pruebas y definiendo un ciclo de vida y los procesos del nivel de gestión de las pruebas, todo esto aplicado a nivel de servicios.

En (Garcia Dominguez & Medina, 2010) se propone el uso de modelos de pruebas en las propias metodologías que sirvan para definir la funcionalidad y la información presente en una SOA. Estos modelos guiarán el uso de varias técnicas para generar casos de prueba y analizar sus resultados.

En (Ramon Rivero Torres, 2010) se propone un modelo que tiene como finalidad proponer principalmente la metodología a usar durante el desarrollo de las pruebas con el fin de ser

usado por cualquier proyecto productivo que siga una línea SOA/BPM. En este artículo se propone el modelo en V como una metodología para desarrollo de este tipo de proyectos, aludiendo a la característica de pruebas paralelas que provee este modelo.

En (Benac Earle, 2014) este trabajo se describe enfoque sistemático para probar aspectos conductuales de Servicios Web que se comunican mediante el formato de datos JSON. Como un componente clave, la herramienta de prueba basada en la propiedad Quviq QuickCheck[12] se utiliza para generar automáticamente un gran número de casos de prueba a partir de una descripción abstracta del comportamiento en servicio en la forma de una máquina de estado finito. La misma descripción de comportamiento también se utiliza para decidir si la ejecución de un caso de prueba tiene éxito o no.

En (Petrova-Antonova, 2015) se propone un marco , llamado Testing as a Service Software Architecture (TASSA) , para pruebas de orquestación de servicios Web, descritas con Business Process Execution Language para Servicios Web (WS - BPEL). Se automatiza un conjunto completo de actividades para el momento de diseñar pruebas funcionales y no funcionales de servicios web. La viabilidad de la estructura se demostró a través de un estudio de caso que demuestra sus características mientras se prueba en un proceso de negocio.

En (Bertolino, 2007) específicamente se detallan seis ejes que contienen los retos que se plantean, y para los que aún no existe solución, estos son:

- La definición del objetivo de las pruebas. Por ejemplo: se buscan fallos, o se necesita decidir si el producto puede ser entregado.
- La selección de los casos de pruebas.
- ¿Qué tan completa y representativa es la prueba, si cubre un porcentaje suficiente del código?
- ¿Qué tipo de prueba es más conveniente aplicar?, o ¿cuál se puede aplicar? (unitaria, de componente, de sistema, de subsistema, de integración), dependiendo del elemento bajo prueba.
- ¿Dónde realizamos la prueba?, en el entorno del desarrollador, en un entorno simulado (virtual), en el escenario de operación (cliente).
- En qué momento del ciclo de vida del producto se ejecuta la prueba.

Estos seis ejes describen un escenario en el que se puede proponer una solución a un subconjunto de ellos, intentando dar respuesta de manera puntual o transversal. Es decir se puede contribuir con un marco teórico para enriquecer el proceso de pruebas a nivel de gestión de procesos y de proyectos y proponer una aplicación práctica en el nivel técnico de un dominio específico. En este sentido existe una necesidad creciente en el dominio de aplicaciones SOA, que integra un nuevo enfoque para el diseño y desarrollo de aplicaciones con las aplicaciones tradicionales.

---

12 Herramienta de la empresa Quviq basada en QuickCheck. QuickCheck es una biblioteca de combinación originalmente escrita en Haskell, diseñado para ayudar en las pruebas de software mediante la generación de casos de prueba para bancos de pruebas.

3. MÉTODO DE PRUEBAS DE INTEGRACIÓN PARA ARQUITECTURAS 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.

Para la realización de este método se procedió a investigar sobre las distintas técnicas, metodologías, herramientas y estudios realizados por otros autores.

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

1. Seguir cada fase de forma ordenada.
2. Mantener comunicación con el equipo de desarrollo sobre las actividades realizadas.

En la Figura 1 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.

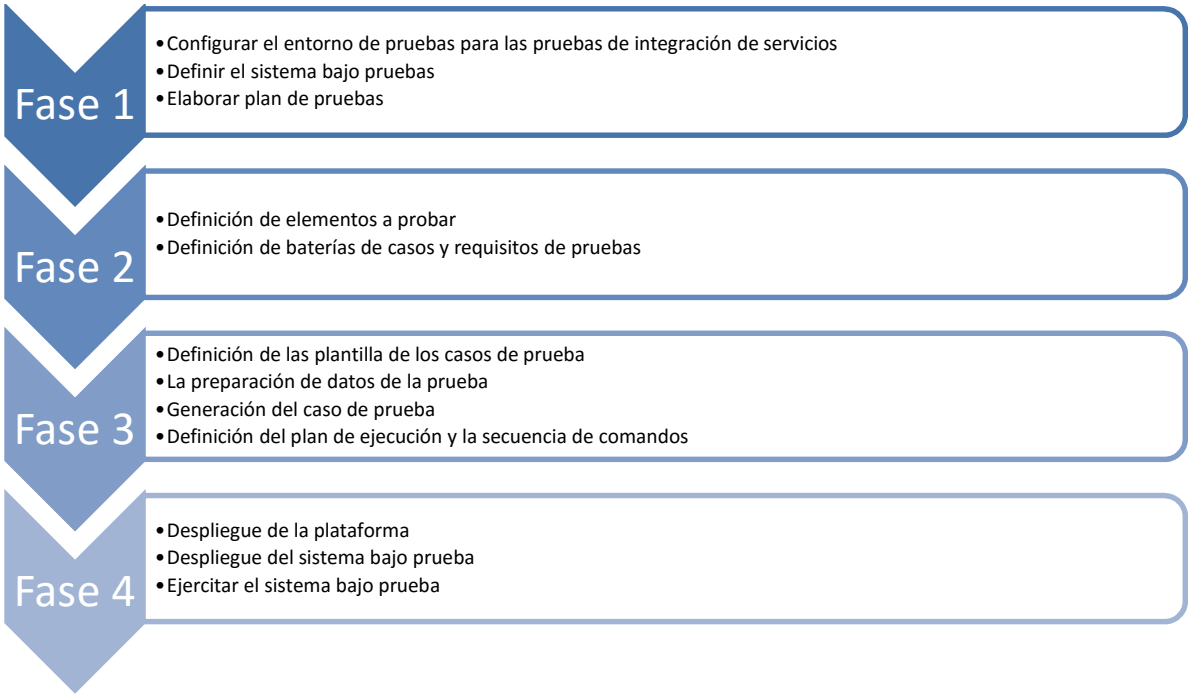

Figura 1 - Fases del método de pruebas

3.1. Descripción de las fases del método

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

3.1.1. 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:

3.1.1.1. 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 basa 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 items para poner a punto el entorno donde se ejercitara el software bajo pruebas:

- Configurar las herramientas que serán utilizadas para este proceso. En esta guia se recomienda el uso de SoapUI[13] para la creación de los casos de prueba en base a los archivos WSDL de las interfaces de los servicios. Para la inyección de datos se utiliza TestMaker PushToTest
- Configurar el espacio de trabajo, definiendo las estructuras de almacenamiento para organizar los artefactos de las pruebas. Esta guia define las siguientes estructuras de almacenamiento minimas:
    - **casos de prueba** almacena los casos de prueba y las configuraciones a realizar.
    - **datos** repositorio para los datos de prueba.
    - **resultados** almacena los resultados de las ejecuciones de los casos de prueba.
    - **wsdl** almacena los archivos de las interfaces de los servicios.
- 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[14], que igualmente debe ser configurada.

3.1.1.2. 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

---

13 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.

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

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 Tabla 1 que contiene la plantilla de Historia del Servicio, la misma se utiliza para cada servicio y se define cada método y requerimiento cubierto.

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.

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			

Tabla 1-Plantilla de Historia del Servicio

3.1.1.3. 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.

Para la creación de este plan de pruebas se propone una estructura que muestra los componentes que hacen parte del mismo:

1. **Introducción**
   1.1. Propósito
   1.2. Entorno
   1.3. Alcance
   1.4. Vision General
2. **Servicios para las pruebas**
3. **Estrategias**
4. **Recursos**
   4.1. Trabajadores
   4.2. Sistema
5. **Hitos del proceso de pruebas**

Los componentes del plan de pruebas pueden cambiar según las necesidades, pero se recomienda tener minimo estos componentes.

3.1.2. 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 y 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:

3.1.2.1. 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. El resultado de esta actividad será la Tabla 2 que contiene el listado de servicios a probar.

Código	Nombre	Descripción	Probar

Código definido en la plantilla de historia del servicio	*Nombre de la historia del servicio*	*Descripción de la historia del servicio*	*S (Si se debe probar), N (Si no se debe probar)*

Tabla 2-Listado de servicios a probar

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 mas se utilicen.

Por ejemplo si se tienen dos servicios, uno para la validación de credenciales y otro para la edición del perfil, y se necesita decidir entre los dos cual se debe probar, la recomedación es probar ambos, pero si es necesario decidir se puede intuir que el de validación de credenciales es mas utilizado que el de edición de los datos del perfil, igualmente tiene un mayor nivel de criticidad, sea por el tema de la seguridad o porque el mismo es un prerrequisito para que el usuario acceda a consumir el resto de servicios.

3.1.2.2. 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 es necesario la realización de 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, 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 2 se observa la ejecución de un falso servicio para un método que realiza la suma de dos números. El Anexo G muestra el proceso para la elaboración de un falso servicio con el software SoapUI para un método que permite sumar dos números.

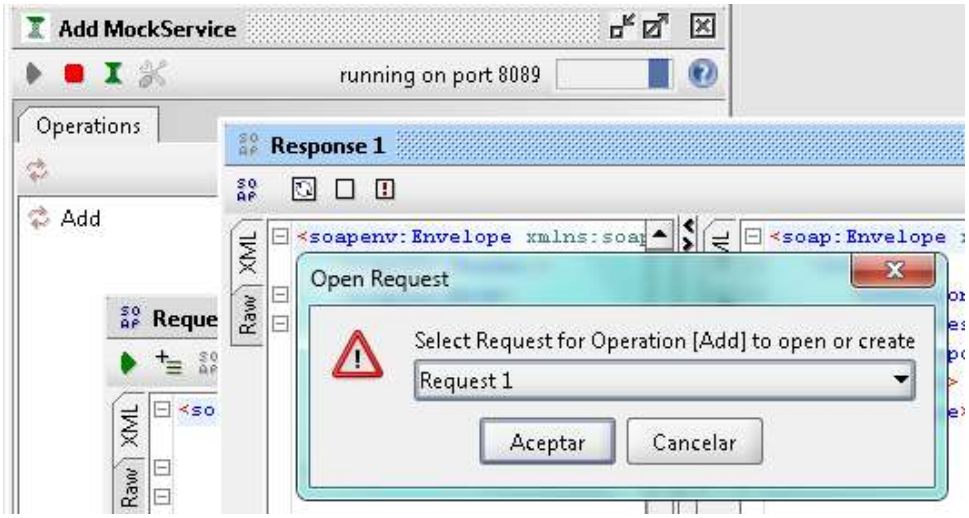

Figura 2 Ejecución del falso servicio en SoapUI

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

Durante esta fase se deben definir los resultados de las pruebas que servirán como oráculo[15] 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.

3.1.3. 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:

3.1.3.1. 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 3. 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.

---

15 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.

```
<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 3 Caso de Prueba en SoapUI

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

3.1.3.2. 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 4 contiene los datos de prueba para un servicio que permite realizar la suma de dos numeros; los valores se encuentran separados por coma, y contiene el valor de los dos numeros, 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 4-Archivo de datos de prueba

Para poder determinar los datos de prueba se deben utilizar los requerimientos, las plantillas de los casos de prueba y las Historias de Servicio.

3.1.3.3. 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 3. 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 5 muestra el esquema de los elementos participantes en el proceso de inyección de datos a la plantilla de prueba a traves de la herramienta PushToTest.

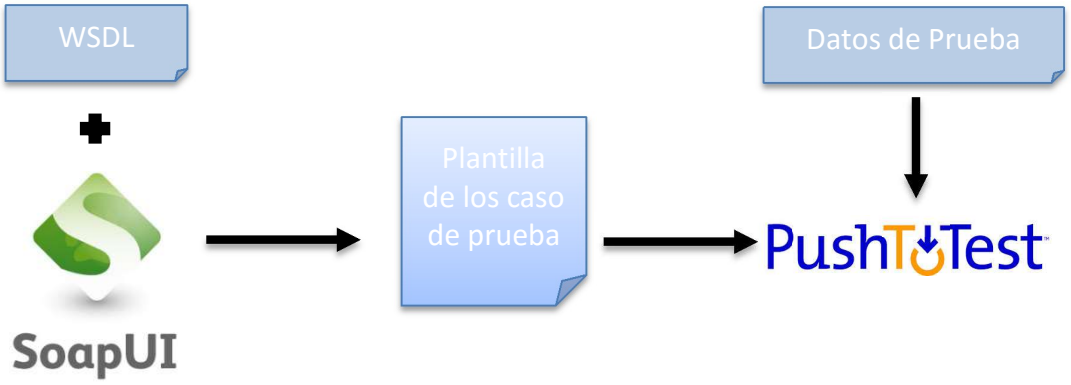

Figura 5-Proceso de inyección de datos

3.1.3.4. 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 6 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 6 Secuencia de comandos en Ant

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.

3.1.4. 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.

3.1.4.1. 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.

3.1.4.2. 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.

3.1.4.3. 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 MÉTODO PROPUESTO

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 Simon Bolivar sede Cúcuta al cual se le aplicarán cada una de las fases del método diseñado.

El contexto del proyecto es la Universidad Simon Bolivar 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.

4.1. Aplicación de las fases del método

A continuación se describirán la aplicación de cada una de las fases de método para pruebas de integración para arquitecturas orientadas a servicios.

4.1.1. 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:

4.1.1.1. Configurar el entorno de pruebas para las pruebas de integración de servicios

En esta actividad se configuran los elementos y las herramientas que serán utilizados para el proceso de pruebas. La Tabla 3 contiene el listado de los elementos de hardware utilizados para el desarrollo de este proyecto y sus respectivas caracteristicas.

Elemento Hardware	Descripción
Servidor	Se utilizo un servidor Virtual sobre VMWare Procesador: Intel Xeon (2C) Memoria: 16 Gb (8 Gb) Sistema Operativo: Windows Server 2012
Equipo de Pruebas	Se utilizo un equipo HP Fisico Procesador: Intel Core i3 Memoria: 4Gb Sistema Operativo: Windows 7

Tabla 3-Hardware utilizado en el proyecto de pruebas

La Tabla 4 muestra la ubicación de los elementos de codigo que componen los servicios desarrollados para las funcionalidades, los cuales exponen las interfaces. Entre estos elementos se encuentra el software bajo prueba, que contiene los servicios desarrollados en lenguaje Java y la ubicación de los archivos WSDL, que corresponden a los archivos que proveen las interfaces.

Elemento	Ubicación

SUT	C:\Tomcat
WSDL Locales	C:\unisimon\wsdl

Tabla 4-Ubicación de elementos del software bajo prueba

La Tabla 5 contiene el listado de elementos que son necesarias para realizar la ejecución del software bajo pruebas. Entre los elementos se encuentra Tomcat que permite desplegar los servicios desarrollados en Java y la base de datos que se encuentra sobre PostgreSQL. Estos elementos deben ser iniciados para que el software bajo prueba se pueda ejecutar.

Elemento	Nombre	Ubicación
Servidores	Tomcat	C:\Tomcat
Base de Datos	Postgresql	C:\Program Files (x86)\PostgreSQL\9.2

Tabla 5-Tecnología necesaria para la ejecución del software bajo prueba

Se instalaron herramientas para la creación y automatización de las pruebas:

- SoapUI v 5.0: se instala la herramienta para la creación de las plantillas de las pruebas de los servicios y la creación de servicios falsos, la herramienta se puede descargar de la página oficial y la instalación en Windows es muy sencilla[16].
- PushToTest TestMaker 7: se instala la herramienta para la automatización de pruebas y la carga de baterías de datos a las pruebas a través de la integración con plantillas de SoapUI[17].
- Ant: se instala para la automatización de tareas y de las pruebas del sistema.

Muchos de estos servicios se desarrollaron en Java, por lo tanto fue necesario instalar una versión de Java Developer Kit.

Se crea una estructura para el almacenamiento de las pruebas en una carpeta llama unisimon ubicada en c:\unisimon. La Tabla 6 contiene la estructura de directorios definidas para la aplicación de pruebas en este proyecto.

Carpeta	Descripción de uso
+ datos	Almacena los datos de prueba que serán utilizados para cada una de las pruebas
+ pruebas	Almacena los archivos que componen las pruebas, entre estos los XML delos archivos SoapUI y los archivos de TestMaker
+ wsdl	Almacena los archivos wsdl que se guardan localmente.

Tabla 6-Estructura de directorios para el almacenamiento de los elementos de las pruebas

16 http://www.soapui.org/

17 http://www.pushtotest.com/

4.1.1.2. Definir el sistema bajo pruebas

Requerimientos del Sistema

La Tabla 7 contiene el listado de los requerimientos de las actividades que debe realizar el sistema a través del consumo de Servicios Web. Esta relación de servicios se basa en la documentación del proyecto, que para este caso corresponde a los casos de uso.

Servicio de Seguridad
- Validaciones credenciales de seguridad.
Servicio Información Académica
- Mostrar información actual del estudiante (se envía el código y se retorna un objeto con la información del estudiante)
- Mostrar Notas Actuales del Estudiante (se envía el código y se retorna un array con las notas)
- Mostrar Histórico de Notas (se envía el código y se retorna un array con el histórico de notas)
- Mostrar Horario del Estudiante (se envía el código y retorna un array con el horario del estudiante)
Servicio Información Financiera
- Mostrar Información Financiera el Estudiante
Realizar pago de matricula
- Realizar pago en línea de la matrícula del Estudiante

Tabla 7-Listado de requerimientos del sistema

Historias de Servicios

Se implementa la plantilla de Historia del servicio propuesto en la Tabla 1. Ver archivo Anexo D con todas las historias de los servicios del proyecto.

4.1.1.3. Elaborar plan de pruebas

Para el proceso de aplicación de las pruebas se elaboró un plan de pruebas que define todo el proceso de aplicación de las pruebas llamado **Plan de Pruebas**. Ver Anexo B.

4.1.2. Fase 2: Diseño y especificación de los casos de prueba de integración de los servicios

4.1.2.1. Definición de elementos a probar

Basado en el tamaño del proyecto y en la criticidad de los servicios, se define que se realizan pruebas de todos los servicios a los cuales accede el sistema. Igualmente esta determinación es

tomada en base a que la cantidad de servicios no es muy alta. En la Tabla 8 se observa el listado de servicios del proyecto y se le da un código basada en una estructura definida para este proyecto.

Código	Nombre	Descripción	Probar
WS01-M01	inicio_pagoV2	Realizar pago de matrícula	S
WS02-M01	verificar_pagoV3	Verificar el estado de un pago	S
WS03-M01	validar_credencial	Verificar las credenciales de autenticación del estudiante	S
WS04-M01	consultar_estudiante	Permite consultar la información básica del estudiante	S
WS04-M02	consultar_notas	Permite consultar las notas actuales de un estudiante	S
WS04-M03	consultar_historico	Permite consultar el histórico de las notas del estudiante	S
WS04-M04	consultar_horario	Permite consultar el horario actual del estudiante	S
WS05-M01	valor_matricula	Verificar el valor del pago de la matricula	S
WS05-M02	consultar_pago	Verificar el valor del estado del pago de la matricula	S

Tabla 8 Listado de servicios a probar

El código WS01-M01 se descompone en dos partes, WS01 corresponde al servicio web, relacionado con el archivo WSDL y la segunda parte M01 corresponde al método del servicio que se desea probar.

Esta codificación viene dada desde la aplicación de la Historia del Servicio. En la Figura 7 se observa la aplicación de la plantilla al primer servicio de este proyecto, y se observa la codificación definida anteriormente.

Información del Servicio

Identificación	WS01			
Nombre del Servicio	Servicio de Zona Pagos			
Descripción del Servicio	Servicio que cumple con las funciones de pagos de ZONAPAGOS para el pago a través de servicios Web			
Naturaleza		Interno	X	Externo
WSDL	https://www.zonapagos.com/ws_inicio_pagov2/Zpagos.asmx?WSDL			
Documento Guía	ZonaVirtual Pasarela de pagos versión Web Service ZP-A-004-INT			

Listado de Métodos

Identificación	WS01-M01					
Nombre	inicio_pagoV2					
Requerimiento	Realizar pago de matrícula					
Criticidad	X	Alto		Medio		Bajo
Parámetros de entrada						
Nombre	**Tipo**	**Observación**				
id_tienda	s:int					
clave	s:string	Opcional				
total_con_iva	s:double					

Figura 7-Aplicación de la plantilla de la historia del servicio

4.1.2.2. Definición de baterías de casos y requisitos de pruebas

Se definen suites de prueba por cada servicio que se desea probar y por cada método a probar se crean baterías de Pruebas, los cuales internamente agrupan los casos de pruebas. Inicialmente se definieron las suites de pruebas que se observan en la Tabla 9 con sus respectivas baterías de pruebas.

Suite de Pruebas	Baterías de Pruebas	Falso	Falso Servicio
Ws01_Suite	Test_Ws01_M01	N	
Ws02_Suite	Test_Ws02_M01	N	
Ws03_Suite	Test_Ws03_M01	S	Mock_Ws03_M01
Ws04_Suite	Test_Ws04_M01	S	Mock_Ws04_M01
	Test_Ws04_M02	S	Mock_Ws04_M02
	Test_Ws04_M03	S	Mock_Ws04_M03
	Test_Ws04_M04	S	Mock_Ws04_M04
Ws05_Suite	Test_Ws05_M01	S	Mock_Ws05_M01
	Test_Ws05_M02	S	Mock_Ws05_M02

Tabla 9 Listado de suite y batería de pruebas

Las suites de prueba se codifican como se observa en la Tabla 9, en el ejemplo Ws01_Suite corresponde a la suite del servicio WS01. Test_Ws01_M01 corresponde a la batería de pruebas para el método WS01-M01 del servicio WS01. La codificación no debe generar complicaciones al momento de aplicar el método, así que la misma queda a disposición del equipo que aplique el método. La codificación se observa en la Figura 8 que muestra el uso de la misma para la creación de los elementos en la herramienta SoapUI.

En el ejercicio se define que los servicios internos contaran con la creación de falsos servicios, ya que los mismos pueden no estar creados, y de esta forma proseguir con la creación de las pruebas. Los falsos servicios también tienen una codificación Mock_Ws03_M01 que para este caso corresponde al falso servicio del método M01 del servicio Ws03.

En la Figura 8 se muestra la estructura de batería de pruebas creada sobre la herramienta de pruebas SoapUI.

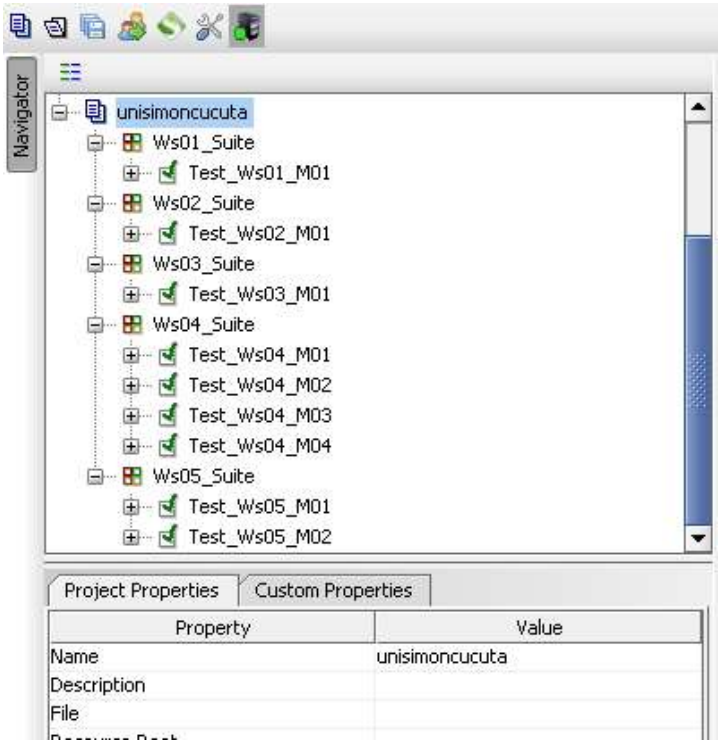

Figura 8-Estructura de batería de pruebas en SoapUI

El archivo creado se almacena en la carpeta de pruebas dentro de la carpeta del proyecto, con el nombre unisimoncucuta-soapui-project.xml.

Durante esta fase se deben definir los resultados de las pruebas que servirán como oráculo de la prueba para la evaluación de las mismas, ya que los mismos pueden ser utilizados en las baterías de pruebas y en los falsos servicios.

4.1.3. Fase 3: Implementación de la Prueba de integración de servicios

4.1.3.1. Definición de las plantilla de los casos de prueba

Para la creación de la estructura del caso de pruebas se utiliza SoapUI, que permite realizar la plantilla del caso de prueba utilizando el archivo WSDL del servicio como se observa en la Figura 9.

Para esta actividad se ubican los archivos WSDL de los servicios a probar, dicha información se encuentra en las historias de servicio. Los archivos pueden ser agregados a la batería de pruebas.

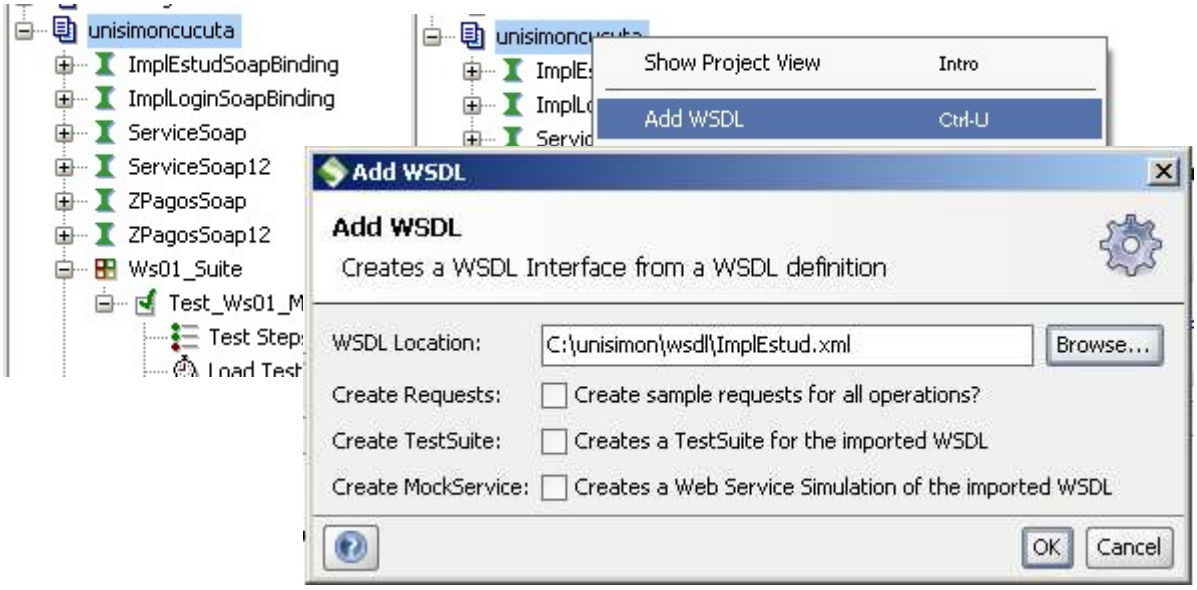

Figura 9 Importación de WSDL

Los archivos WSDL son importados y SoapUI permite crear las peticiones de las operaciones del servicio si se desea, así como la suite de pruebas y los falsos servicios. Para este caso, primero se importaron las WSDL de los servicios sin crear las peticiones.

El siguiente paso después de haber agregado el WSDL es crear la petición, acción que genera la plantilla de la petición como se observa en la Figura 10 para el método consultar_historico, el cual recibe el código del estudiante como parámetro.

Figura 10 Plantilla de petición de caso consultar_historico

En esta actividad se generaron las peticiones para cada servicio a probar como se muestra en la Figura 11. Por facilidad a todos se les llamo Request 1.

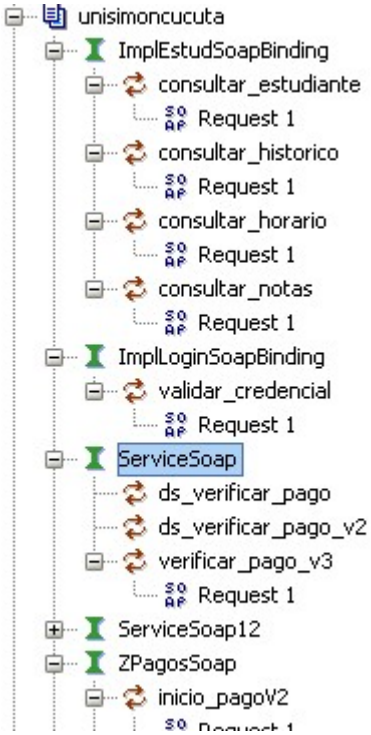

Figura 11-Peticiones de los casos de prueba

Igualmente en cada petición se generan las plantillas de cada servicio. Como se observa en la Figura 12 la herramienta SoapUI genera la plantilla con signos de "?" en los parámetros de entrada para el servicio ImpLogin.

Figura 12 Plantilla caso de prueba validar_credencial

La Figura 13 muestra la plantilla del caso de prueba para el servicio ws_verificar_pago generado en la herramienta SoapUI.

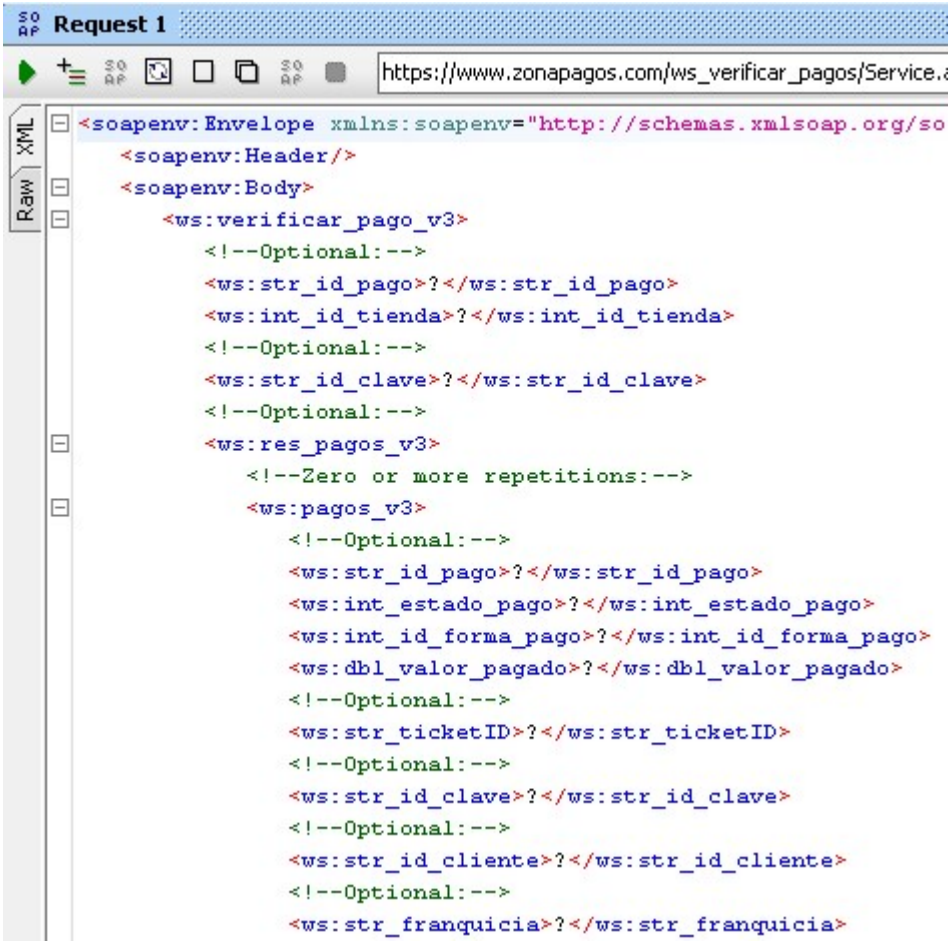

Figura 13 Plantilla caso de prueba verificar_pago

Paso seguido se definieron las aserciones o validaciones para cada una de las pruebas o pasos. En SoapUI hay diversos tipos de pruebas, pero para el tema de integración nos basaremos en realizar las pruebas de invocaciones sobre SOAP (Test Request). Hay varios tipos de aserciones, para este caso vamos a manejar aserciones según se hayan definido en la especificación como se observa en la Tabla 10.

Baterías de Pruebas	Caso de Prueba	SOAP	XPATH	SLA
Test_Ws01_M01	inicio_pagoV2	X	X	X
Test_Ws02_M01	verificar_pagoV3	X	X	X
Test_Ws03_M01	validar_credencial	X	X	
Test_Ws04_M01	consultar_estudiante	X	X	
Test_Ws04_M02	consultar_notas	X	X	

Test_Ws04_M03	consultar_historico	X	X	
Test_Ws04_M04	consultar_horario	X	X	
Test_Ws05_M01	valor_matricula	X	X	
Test_Ws05_M02	consultar_pago	X	X	

Tabla 10 Validaciones a implementar en los casos de prueba

Para facilidad de la validación, se aplican a este caso las pruebas más comunes:

SOAP: Corresponde a la prueba SOAP Response, la cual valida que la última respuesta recibida corresponde a un mensaje SOAP. Con esta aserción se pretende validar que el mensaje de respuesta es SOAP y se valida que el consumo del servicio sea correcto.

XPATH: Corresponde a la prueba Xpath Match, la cual utiliza una propiedad Xpath para obtener el contenido de una propiedad objetivo y compararlo con un valor determinado. Con esta aserción comparamos los datos que se obtienen en la respuesta con un valor especifico, que puede ser cedula, nombre, etc. Así determinamos que la respuesta recibida realmente contiene los datos que queremos.

SLA: Corresponde a la prueba Response SLA, la cual valida la ejecución en un determinado tiempo. Con esta aserción validaremos el tiempo de los servicios que son externos a la aplicación, solicitando un acuerdo de servicio de un respectivo tiempo.

En la Figura 14 se puede observar las validaciones realizadas sobre una de las pruebas. En ese caso se observa la validación SOAP Response y la validación XPath Match.

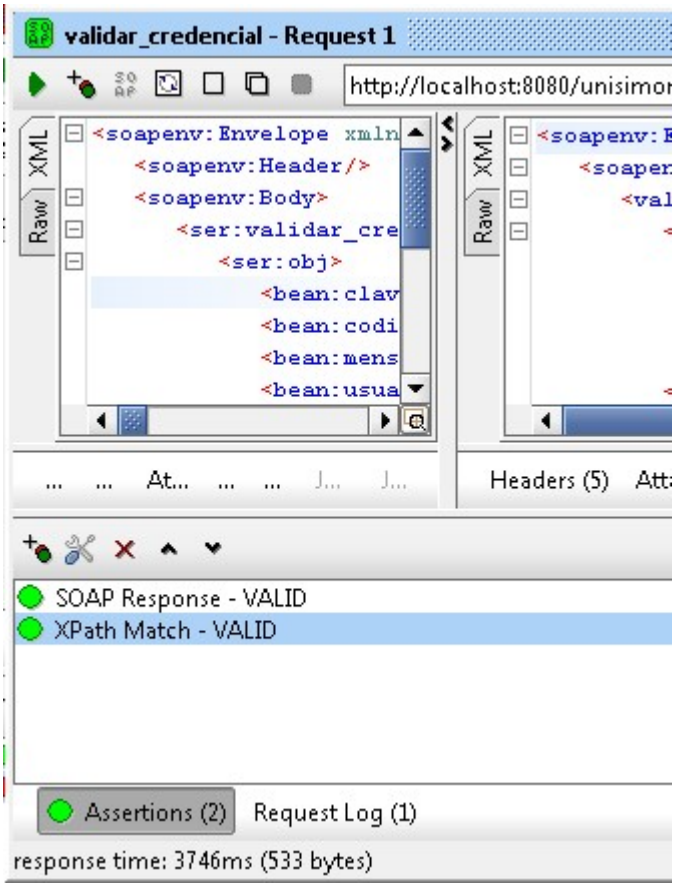

Figura 14-Validaciones realizadas sobre casos de prueba

4.1.3.2. Preparación de los datos de prueba

Para esta parte se tienen en cuenta las historias de los servicios, las plantillas de los casos de pruebas y las aserciones aplicadas a los mismos. Se crearon los archivos de datos, y cada uno se codifico con el nombre del caso de prueba que utilizaba y se almacenaron en la carpeta de datos.

```
usuario,clave,codigo,mensaje
c_angarita,unisimon,201422511166,Usuario Existente
a_abril,unisimon,201422511165,Usuario Existente
c_angarita,unisimon2,2201422511166,Usuario Erroneo
```

Figura 15 Ejemplo de archivo de datos para caso de prueba

En la Figura 15 se presenta los datos de prueba basados en los requisitos del método validar_credencial. Por ejemplo para el caso anterior, se pasa el valor Usuario Existente como respuesta al servicio, para determinar si el usuario existe en la base de datos y las credenciales de acceso son correctas.

Para cada caso de prueba se definen los datos de prueba que se van a utilizar y se almacenan en la respectiva carpeta de datos de prueba.

4.1.3.3. Generación del caso de prueba

Para cada uno de los casos de prueba se modificó la plantilla, adicionando campos de las propiedades que permiten transferir datos hacia los casos de prueba. En este paso se crearon inicialmente las propiedades, dando nombres en base a los valores que se tienen en el archivo de datos de prueba.

Las propiedades en SoapUI se crearon y luego se asociaron en la plantilla como se observa en la Figura 16, reemplazando los valores de interrogación (?) por el parámetro correspondiente de la lista de propiedades:

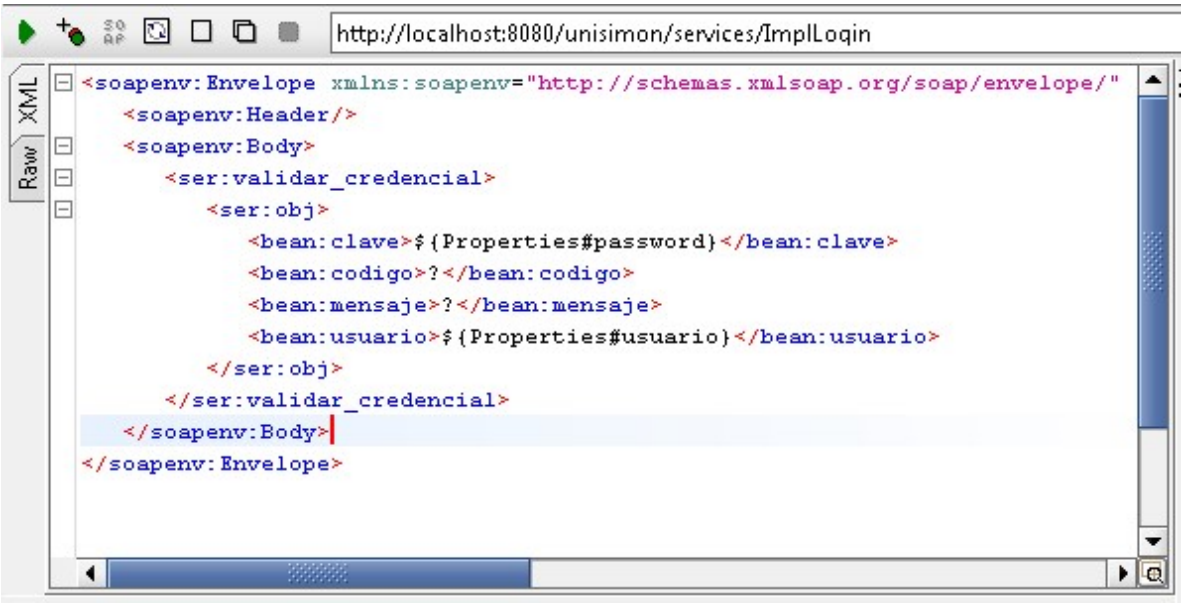

Figura 16 Paso de valores del archivo de propiedades a la plantilla del caso de prueba

Este proceso de asignación de valores de la lista de propiedades a los parámetros de los servicios se debe realizar para todas las pruebas.

En esta fase también se relacionaron los archivos de datos a través de la herramienta de carga de datos PushToTest TestMaker, sobre la cual se realiza un escenario que me permite vincular el archivo guardado desde SoapUI, que se encuentra en la carpeta de pruebas.

Por facilidad se creó un único escenario, y en el mismo se agregaron todas las pruebas vinculadas con los datos de prueba. La Figura 17 muestra el escenario de pruebas creado para el proyecto.

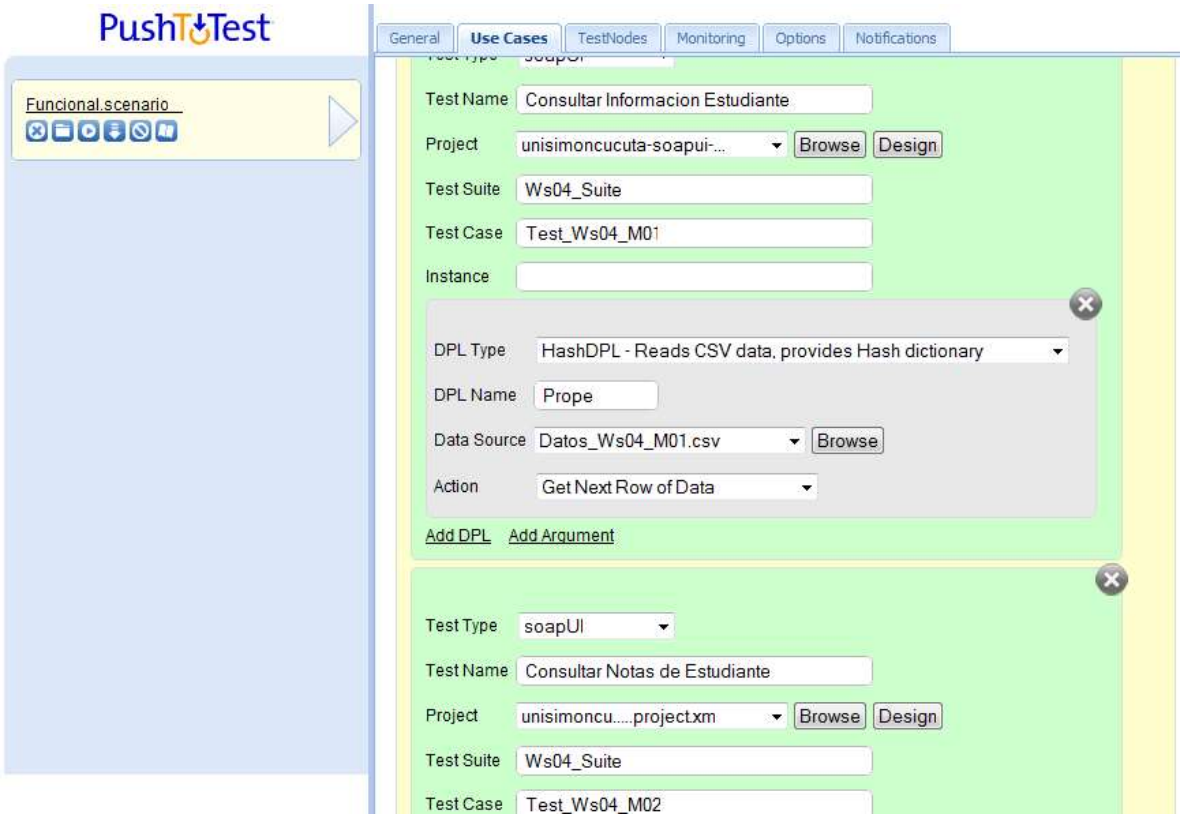

Figura 17 Escenario de pruebas en TestMaker

4.1.3.4. Definición del plan de ejecución y secuencia de comandos

En este paso se definieron las secuencias de comandos Ant para poder realizar todas las actividades desde iniciar los ambientes de desarrollo y pruebas. La Figura 18 contiene parte de las sentencias Ant creadas para poder realizar la ejecución del proyecto.

```
<project  name="test" basedir="." default="tomcat-stop">

   <property name="project" value="./PruebaAlicante-soapui-project.xml" />
   <!-- Define las variables del ambiente de pruebas -->
   <property name="tomcat.home" value="C:\tomcat" />
   <property name="postgres.home" value=" C:\Program Files (x86)\PostgreSQL\9.2" />

   <!-- Define la raiz de la carpeta de pruebas -->
   <property name="base" value="C:\unisimon" />

   <property name="wsdl" value="${src}\wsdl" />
   <property name="datos" value="${src}\datos" />
   <property name="pruebas" value="${src}\pruebas" />

   <!-- Define las herramientas configuradas para la realización de las pruebas -->
   <property name="dirtestmaker" value="C:\Program Files\PushToTest_TestMaker" />
   <property name="dirsoapui" value="C:\Program Files (x86)\SmartBear\SoapUI-5.0.0" />

   <property name="os" value="Windows 7" />

   <target name="tomcat-start">
      <java classname="org.apache.catalina.startup.Bootstrap" fork="true">
         <classpath path="${tomcat.home}/bin/bootstrap.jar:${tomcat.home}/bin/tomcat-juli.jar" />
         <jvmarg value="-Dcatalina.home=${tomcat.home}"/>
```

Figura 18 Definición de comandos Ant para ejecución de pruebas

En el ejemplo se separaron las ejecuciones en tres archivos, el primero en las instrucciones para el despliegue de la plataforma, que en este caso consiste en desplegar la base de datos y el servidor de aplicaciones.

Adicional se creó el archivo para iniciar el ambiente bajo pruebas, lo cual incluye todos los elementos necesarios para poder aplicar las pruebas, para lo cual en este caso fue necesario desplegar todas las dependencias de TestMaker.

4.1.4. Fase 4: Ejecución de la Prueba y análisis de resultado

Durante esta fase se realizo la ejecucion del ambiente pruebas a traves de los comandos Ant que fueron creados en las fases anteriores.

4.1.4.1. Despliegue de la plataforma

En esta etapa se ejecutaron las sentencias para desplegar los elementos de la plataforma:

- Base de datos: se despliega PostgreSQL 9.2, el cual se encuentra instalado sobre Windows.
- Servidor de Aplicaciones: se despliega TOMCAT.

Para esto se utilizan secuencias ANT para desplegar estos elementos.

4.1.4.2. Despliegue del sistema bajo prueba

Para esta actividad se ejecutaron todas las herramientas necesarias para ejecutar las pruebas, entre las que estan los componentes del TestMaker. Para esto se utilizaron sentencias ANT. Dentro de los servicios que utiliza TestMaker se encuentra una base de datos y un conjunto de servicios de monitoreo.

4.1.4.3. Ejercitar el sistema bajo prueba

En esta actividad se ejecuto la sentencia que permite ejecutar el escenario de prueba de TestMaker que incluye la vinculación de las plantillas de prueba SoapUI. Las plantillas se ejecutaron a traves de la consola utilizando sentencias ANT, pero tambien se puede ejecutar a traves de sistemas de integración continua que permiten ejecutarlas. Para el caso de este ejercicio se utilizo Hudson, pero pueden utilizar Jenkins u otro.

5. CONCLUSIONES Y RESULTADOS

5.1. CONCLUSIONES

En el presente trabajo se aborda la temática relacionada con las pruebas de software en aplicaciones orientadas a servicios, donde se toman los modelos tradicionales de prueba y se aplican a estas nuevas arquitecturas de desarrollo. Como resultado de la investigación realizada se han obtenido las siguientes 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.

5.2. RESULTADOS

5.2.1. Estado del Arte

Los estudios realizados sobre las pruebas de software son extensos, lo cual demuestra que es una disciplina estructurada con métodos, técnicas y procesos definidos, que con la aparición de nuevas arquitecturas, ha comenzado a adaptarse. Referente a las pruebas en arquitecturas orientadas a servicios, se cuenta con diversos documentos que permiten la aplicación de pruebas en diversos niveles, especialmente en el nivel de unidad o servicio, ya que se ocupa de validar la funcionalidad de la unidad de negocio o del procedimiento que es invocado al consumir el servicio. En relación al nivel de integración en arquitecturas orientadas a servicios se encontraron algunos estudios correspondientes a los últimos años, así como documentos comerciales de algunas empresas que pretenden apoyar la formalización de las pruebas sobre el nivel de integración en esta arquitectura. Cabe resaltar que todos han recalcado la importancia de contar con la disposición de servicios, y como esta arquitectura supone conectarnos con terceros, muchos recurren a la creación de falsos servicios como solución a dar continuidad al proceso de pruebas.

5.2.2. El método

Se propone un método para pruebas de integración en desarrollos con arquitecturas orientadas a servicios, disponible para que el personal encargado del proceso de pruebas pueda planear y ejecutar las pruebas de integración.

El método se compone de 4 fases que van desde el diseño hasta la ejecución, con las respectivas actividades correspondientes y se proponen un conjunto de herramientas que sirven para configurar un entorno que permita la ejecución del mismo.

5.2.3. Validación del método

La validación del método se realizó con el apoyo del departamento de sistemas de la Universidad Simón Bolívar sede Cúcuta y retorna un conjunto de resultados tanto positivos como negativos y se muestran a continuación.

Ventajas

- El método permitió a los desarrolladores mejorar el proceso de desarrollo y ayudo a eliminar la dependencia entre los responsables del desarrollo de los servicios y los responsables del desarrollo de la aplicación; a través de la implementación de los servicios falsos (Mock Services) los encargados del desarrollo de la aplicación obtienen respuesta de los web services así los mismos no se encuentren disponibles, con este proceso no dependo de la terminación del servicio para comenzar con el desarrollo de la aplicación.
- La automatización de las pruebas de integración a través de herramientas de software permitió mejorar el proceso de desarrollo y ayudo a que la validación de los cambios no fuesen tan complicada, debido a que fácilmente se podían volver a ejecutar los scripts para poder validar la validez de las pruebas.

Desventajas

- Gestionar el proceso de aplicación de pruebas de integración a través de este método conlleva que algunos desarrolladores deban dedicar un poco más de tiempo a estas funciones de diseño y realización de las pruebas. Para el caso del equipo de la

Universidad Simón Bolívar no se contaba con este perfil y el proceso se apoyó en un desarrollador.

- Algunos desarrolladores no se encontraban conformes con la aplicación del método, debido a que no se cuenta con una política de aplicación de pruebas completa y documentada.
- Hay que articular diversas herramientas para la automatización de las pruebas y esto conlleva un trabajo adicional al diseño y ejecución de las pruebas. Para este proceso la configuración de las herramientas gasto más tiempo de lo esperado.

5.2.4. Participación en Eventos

En el marco de la investigación se logró participar con la Ponencia titulada "Estado del arte de las pruebas de software", en la VI Semana de Ciencia, Tecnología e Innovación, organizado por la Vicerrectoría de Investigación de la Universidad de Francisco de Paula Santander, San José de Cúcuta –Norte de Santander, 19, 20 y 21 de Octubre de 2009. Ver Anexo F.

6. BIBLOGRAFÍA

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

Benac Earle, C. (2014). Jsongen: a quickcheck based library for testing JSON web services. págs. 33-41.

Bertolino, A. (2007). Software Testing Research: Achievements, Challenges, Dreams. En IEEE (Ed.), *Future of Software Enginnering.*

Bogdan, & Taylor. (1987). *Introducción a los métodos cualitativos* (Segunda ed.). Paidós.

Canfora G., D. P. (2006). Testing services and service-centric systems: challenges and opportunities. *8*(2), 10-17.

Carazo, P. C. (2006). El método de estudio de caso; Estrategia metodologica de la investigación cientifica. *pensamiento & gestión, 20. Universidad del Norte*, 165-193.

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.

Dueñas, J. C., & Garcia, B. (2009). Modelo de Pruebas de Software en el Desarrollo de Aplicaciones Orientadas a Servicios. *Libro de Actas de las VIII Jornadas de Ingeniería Telemática*, (págs. 223-230).

Dustdar, S. (2004). Testing of Service Oriented Architectures - A practical approach. *Lecture Notes in Computer Science*, 97-109.

Estrada Martinez, M. F. (2010). *FRAMEWORK PARA EL PROCESO DE TESTING.*

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

Garcia Dominguez, A., & Medina, B. (2010). Hacia la Integración de Técnicas de Pruebas en Metodologías Dirigidas por Modelos para SOA. *Novática*.

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

Gomez, M. M. (2006). *Introducción a la Metodología de la Investigación Científica* (Primera ed.). Cordoba, Argentina: Brujas.

Gonzalez, L. (2009). Método para generar casos de prueba funcional en el desarrollo de software.

Grinell, R. (1997). *Social work research & evaluation: Quantitative and qualitative approaches. (5a. Edicion). . Peacock Publishers.* (Quinta ed.). (Illinois, Ed.) Itaca, E. E: Peacock Publishers.

Gutierrez, J., & Escalona, M. (2006). MODELOS DE PRUEBAS PARA PRUEBAS DEL SISTEMA. *XV Jornadas de Ingeniería del Software y Bases de Datos .* Barcelona.

Hernández, S. R., Fernández, C. C., & Baptista, L. P. (2010). *Metodología de la Investigación* (Quinta ed.). México, México: McGraw - Hill Interamericana.

Hernandez, S. V. (2002). *Esquema de caracterización para la selección de técnicas de pruebas de software.* Tesis Doctoral, Universidad Politecnica de Madrid.

Heumann, J. (s.f.). *Generating Test Cases From Use Cases.* Obtenido de The Rational Edge: http://www.ibm.com/developerworks/rational/library/content/RationalEdge/jun01/GeneratingTestCasesFromUseCasesJune01.pdf

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

IEEE-CS. (2004). *Guide to the Software Engineering Body of knowledge.*

Jing G., Y. L. (2008). A Model of Third-Party Integration Testing Process for Foundation Software Platform. *The 9th International COnference for Young Computer Scientists*, 1199-1204.

Johanna Rojas Rojas, E. J. (2007). *Investigacion sobre estado del arte en diseno y aplicacion de Pruebas de Software.* Bogota. Recuperado el 09 de 04 de 2009, de [9]. http://www.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node1.html

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

Laguado, G. Y. (2007). *Caracterizacion de los Servicios Web y las Tecnologias actualmente vinculadas a su desarrollo SOAP, WSDL, UDDI.* Cucuta: Universidad Francisco de Paula Santander.

Lenz Ch., C.-O. J. (2007). Model driven testing of soa-based software. *Proceeding of the SEMSOA Workshop 2007 on Software Engineering Methods for Service-Oriented Architecture, 244*.

Londesbrough. (April de 2008). A Test Process for all Lifecycles. *Software Testing Verifiation and Validation Workshop ICSTW 08*.

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

Moreno Ceballos, G. R., & Morales Oliva, A. (2008). Retos impuestos por Arquitectura Orientada a Servicios al concebir una estrategia de pruebas. *Serie Científica.*

Myers, G. J. (1979). *The Art of Software Testing.*

OMG, UML Testing Profile. (2010).

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

Petrova-Antonova, D. (2015). Tassa - Testing Framework for web service orchestrations. *Proceedings of the 10th International Workshop on Automation of Software Test*, págs. 8-12.

Project Management Institute. (2013). *A Guide to the Project Management Body of Knowledge (PMBOK® Guide).* Fifth Edition (Quinta ed.).

Ramon Rivero Torres, E. O. (2010). Propuesta de un modelo de pruebas para una Arquitectura Orientada a Servicios. *Eighth LACCEI Latin American and Caribbean Conference for Engineering and Technology (LACCEI'2010) "Innovation and Development for the Americas"*.

Ribarov L., M. I. (2007). Testing in Service Oriented World. *Intenational Conference on Information Technologies* , *1.* Bulgaria.

Riebisch. (2003). UML-Based Statistical Test Case Generation. *NetObjectDays on Objects, Components, Architectures, Services, and Applications for a Networked World.*

Rodríguez Peñuelas, M. A. (2005). *Metodología de investigación. Material de curso de seminario de tesis de la Maestría en Impuestos.* Universidad Autónoma de Sinaloa, Facultad de Contaduría y Administración, México.

Sampieri, R. H., Collado, C. F., & Lucio., P. B. (2010). *Metodología de la Investigación* (Quinta ed.). México, D. F.: McGraw-Hill, Interamericana.

Serna, E., & Arango, F. (2011). Prueba del software: más que una fase en el ciclo de vida. (U. d. Andes, Ed.) *Revista de Ingeniería*(35), 34-40.

Silverman, D. (1995). *Interpreting qualitative data. Methods for analysing talk, text and interaction.* Sage, Londres.

Sinha A., P. A. (17 de 07 de 2006). Model-based functional conformance testing of web services operating on persistent data. (ACM, Ed.) *Proceeding of the 2006 Worksho on Testing, Analysis, and Verification of Web Services and Applications*, págs. 17-22.

Sinha, A. (2006). Model-based functional conformance testing of web services operating on persistent data. *Proceedings of the 2006 workshop on Testing, analysis, and verification of web services and applications*, 17-22.

Torres, A., & Escalona, M. (2009). Oráculos de prueba - Un planteamiento heurístico de apoyo a decisión . *Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos.*

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

Tsai, W.-T., & Jerry, G. (2008). Testability of Software in Service-Oriented Architecture.

Yoon H., J. E. (2008). Building Test Steps for SOA Service Orchestration in Web Service Testing Tools. *Proceedings of the 2nd International Conference on Ubiquitous Information Management and Communication*, (págs. 555-557). Suwon, Korea.

7. ANEXOS

Anexo A – Documento de Casos de Uso

Anexo B – Plan de Pruebas

Anexo C – Plantilla de Historia de Servicios

Anexo D – Aplicación de la Plantilla de Historia de Servicios

Anexo E – Plan del Proyecto

Anexo F – Articulo Estado del Arte de las Pruebas de Software

Anexo G – Certificación de Ponencia

Anexo H – Creación de Servicios de Prueba con la herramienta SoapUI