AUTOMATIZACIÓN DE PRUEBAS DE SOFTWARE WEB BASADA EN REGLAS DE NEGOCIO

JUAN CAMILO SALAZAR RODRIGUEZ

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
FACULTAD DE INGENIERÍA
ESPECIALIZACIÓN EN INGENIERÍA DE SOFTWARE
BOGOTÁ D.C.
2016

Índice general

Índice de figuras

Índice de cuadros

Introducción

El presente trabajo de investigación pretende explorar una solución al problema del mantenimiento de pruebas automatizadas, a través de la integración de un sistema de administración de reglas de negocio con un framework de Automatización de Pruebas. La importancia de realizar automatización de pruebas en proyectos de software nuevos o existentes está dada por los elevados costos asociados a la fase de pruebas de un proyecto, que en algunas ocasiones ha llegado a superar el 50 % del costo total.

Por otra parte, las empresas que han intentado realizar una implementación de automatización de pruebas se enfrentan al hecho de que posteriormente el mantenimiento de los scripts de prueba tienen un alto costo en su mantenimiento, además de que para su construcción es necesario contar con personal con conocimiento y experiencia en el manejo de los frameworks que permiten la automatización.

Para poder dar solución al problema planteado, se identifica las reglas de negocio como una posibilidad para reducir la manteniblidad de los scripts de prueba, dados sus antecedentes respecto a la reducción de la complejidad en la creación de nuevas aplicaciones, que ha permitido que personal de negocio pueda administrar cambios en sus sistemas de información sin requerir conocimiento técnico previo.

Por tanto, se investigará la integración mencionada, y se aplicará a un software – caso de estudio para corroborar si efectivamente puede llegar a disminuir el mantenimiento asociado a los scripts de pruebas.

PARTE I

CONCEPTUALIZACIÓN DE LA INVESTIGACIÓN

CAPÍTULO 1
Descripción de la Investigación

1.1. Planteamiento/Identificación del Problema

Actualmente, las empresas invierten más del 50 % del costo total de un proyecto en la ejecución de pruebas Alégroth, Feldt y Kolström, 2016, lo cual ha despertado interés en cómo mejorar el proceso global de pruebas a través de la automatización. Sin embargo, al intentar implementar iniciativas de automatización de pruebas, las organizaciones se enfrentan a:

1. Un incremento de los costos del proyecto Karhu, Repo, Taipale y Smolander, 2009, originado por la inversión adicional en infraestructura tecnológica y en contratación o entrenamiento de personal con conocimiento en automatización

2. Actividades adicionales durante la fase de operación del proyecto, enfocadas al mantenimiento de los scripts de automatización.

Con el fin de disminuir en parte el incremento en costos y tiempos en la automatización, se plantea como posible solución el manejo de las reglas de negocio implícitas en los scripts por parte del personal de negocio y/o los analistas/testers, mediante un motor que centralice la administración de las reglas de negocio.

1.2. Objetivos

1.2.1. Objetivo General

Implementar un prototipo de automatización de pruebas, que mediante la interacción entre un motor de reglas de negocio y un framework de automatización de pruebas, permita comprobar la viabilidad de aplicar dicha integración en los proyectos de pruebas automáticas.

1.2.2. Objetivos Específicos

1. Identificar las herramientas y técnicas existentes en la actualidad para la implementación de motores de reglas de negocio, por medio de una investigación técnica de las opciones disponibles, para tenerlas en cuenta al momento de selección y despliegue de un motor de reglas de negocio.

2. Identificar las características de una implementación correcta de la automatización de pruebas, por medio de la investigación de las buenas prácticas en este campo, para asegurar la implementación exitosa del proceso de automatización.

3. Implementar un prototipo de software para automatizar las pruebas del caso de estudio "Administrador de Encuesta Docente - Componente Confirmación Clases", por medio de una integración entre un motor de reglas de negocio y un framework de automatización de pruebas, para comprobar la validez de la integración.

1.3. Justificación del trabajo/investigación

La ejecución de pruebas sobre software nuevo o prefabricado ha sido un reto para las organizaciones, ya que por lo general implica la contratación de personal adicional y disponer de un tiempo considerable para lograr un cubrimiento de pruebas óptimo. Dicha situación puede volverse compleja en organizaciones que implementan soluciones tipo ERP o CRM, dónde en varias ocasiones realizan desarrollos a la medida y/o configuraciones sobe las soluciones adquiridas, que pueden llegar a afectar el funcionamiento normal de dichas plataformas.

Frente a dicho problema, algunas organizaciones han optado por implementar scripts de automatización de pruebas lo cual, aunque al principio arrojan resultados positivos, al largo plazo implica alta mantenibilidad de los desarrollos realizados para dicho fin. De hecho, uno de los impedimentos para implementar la automatización es el bajo reuso e inflexibilidad de los scripts Zhu, Zhou, Li y Gao, 2008, lo cual origina contratación de ingenieros especializados y por tanto incrementa el costo de los proyectos.

El problema de la mantenibilidad en la automatización de pruebas se debe en parte a que no es posible tomar los cambios en las organizaciones que dan lugar a cambios o creación de nuevos casos de prueba y ajustarlos en los scripts de automatización. El funcionamiento lógico de las pruebas, al igual que los resultados y posibles cambios que pueden suceder en los casos de prueba deben tener cierta interpretación desde el punto de vista del negocio, de ahí que es valioso que dicho conocimiento pueda residir desde un inicio en la(s) persona(s) que desarrollen los scripts y de esta forma realizar ágilmente los ajustes correspondientes.

Para lograr que el conocimiento de negocio no afecte la mantenibilidad de los scripts de pruebas, es posible para las actividades de automatización de scripts adoptar el uso de reglas de negocio, ya utilizado en otras disciplinas que lograron acoplar los cambios que suceden en las organizaciones (y que originan cambios en los casos de pruebas) de una manera ágil. Una regla de negocio es "una declaración que define o restringe algún aspecto del negocio" The Business Rules Group, 2000, que puede entender las personas de negocio y que a su vez cuentan con suficientes detalles para su implementación para funcionales y técnicos d. Jesus y d. Melo, 2015. Además, una vez definidas pueden ser administradas a través de un motor de reglas de negocio, que se puede sincronizar con el código o software cada vez que ocurran cambios en dichas reglas.

De esta manera, esta investigación pretende explorar la automatización de pruebas vista desde la perspectiva de reglas de negocio, dirigido a identificar si es una posible solución al problema de mantenibilidad mencionado anteriormente.

1.4. Hipótesis

Es posible realizar una integración entre un motor de reglas de negocio con un framework de automatización de pruebas.

1.5. Metodología de la Investigación

La metodología a emplear en la presente investigacion (o proyecto de grado) será de tipo teórico-práctica. Por tanto, el desarrollo del proyecto se agrupará en tres grandes fases:

1. *Investigación Teórica*: en esta fase se realizará una revisión bibliográfica y de fuentes electrónicas sobre los ejes conceptuales fundamentales del proyecto (Reglas de Negocio y Automatización de Pruebas).

   Para el desarrollo de esta fase, se utilizarán fuentes secundarias, las cuales servirán como soporte teórico y documental sobre el uso del BRMS o framework de automatización de pruebas escogido para el prototipo. Dichas fuentes procederán de:

   - Manuales de uso del BRMS/Framework Automatización escogidos
   - Libros
   - Artículos que describan la forma de uso de BRMS/Frameworks Automatización

2. *Investigación Práctica*: en esta fase, se realizarán actividades relacionadas con:

   - La selección del motor de reglas de negocio y el framework de automatización de preubas
   - La creación de una arquitectura candidata para la integración del motor de reglas y el framework de automatización seleccionado
   - El análisis del Software caso de estudio sobre el cual se va a aplicar la arquitectura propuesta.
   - La aplicación de la arquitecutra candidata sobre el Software caso de estudio (es decir, la construcción del prototipo)

3. *Resultados y Conclusiones*: en esta fase se recolectarán los resultados de aplicar la arquitectura candidata planteada sobre el Software caso de estudio, y se analizarán a la luz de los objetivos planteados en el proyecto.

1.6. Organización del trabajo de grado

El presente trabajo de grado se encuentra organizado de la siguiente manera:

- Se iniciará con un capítulo en dónde se explica el enfoque de reglas de negocio (Business Rules Approach), mostrando sus beneficios al momento de mejorar el mantenimiento del software.

- Como otro eje central de la hipótesis de este trabajo, será necesario exponer la forma en la cual se maneja la automatización de pruebas de Software, al igual que las mejores prácticas que existen para su implementación.
- De acuerdo a las características expuestas en los dos capítulos previos, se expondrá en el Capítulo 4 la descripción general del modelo de automatización de pruebas que pretende implementar este trabajo de grado, al igual que las condiciones de aplicación.
- Por último, se expondrá los resultados arrojados por la implementación del prototipo de automatización de pruebas, y cómo la integración con reglas de negocio facilitó la mantenibilidad de los scripts de prueba.

PARTE II

DESARROLLO DE LA INVESTIGACIÓN

CAPÍTULO 2
Reglas de Negocio

2.1. Introducción

Los seres humanos nos distinguimos de otros seres vivos por nuestra habilidad de tomar decisiones. Dichas decisiones están guiadas por hechos que suceden en nuestro contexto, y reglas que determinan nuestro actuar, ya sea a nivel moral, social o personal. De igual manera, las decisiones que toma una empresa son guiadas por los hechos que suceden en su entorno competitivo, y por el conjunto de políticas organizacionales que dirigen su quehacer diario.

A partir de lo anterior, para que una empresa logre tomar decisiones acertadas y de calidad, es necesario que cuente con un sólido conjunto de reglas de negocio von Halle, 2001. Una regla de negocio se define como *"una declaración que define o restringe algún aspecto del negocio"* The Business Rules Group, 2000, que *"influencia el comportamiento colectivo de los miembros de una organización o sus sistemas de información"* von Halle, 2001.

Una regla de negocio se considera que es distinta a las leyes naturales, físicas, regulatorias, estándares externos o buenas prácticas. Dado que las reglas de negocio se consideran como optativas en el sentido que un negocio puede decidir prescindir de ellas Witt, 2012, existen algunas reglas que, aunque una empresa u organización puede decidir no ejecutar, implicarían riesgos legales, financieros y de optimización. Por tanto, son reglas que no deberían ser opcionales.

Sin embargo, que un grupo de compañías se vean regidas por un mismo conjunto de reglas, leyes o regulaciones no implica que todas posean las mismas reglas de negocio. De hecho, un factor diferenciador entre organizaciones es el hecho de aplicar su propio conjunto de reglas que ayuden a guiar su toma de decisiones tácticas y estratégicas, diferenciándose así de la competencia Loshin, 2013. Por tanto, además de definir reglas de negocio es importante identificar su motivación Witt, 2012, con el fin de que desde una perspectiva organizacional se pueda medir su impacto visto desde varias perspectivas.

2.2. Importancia de las Reglas de Negocio

De acuerdo al tipo de motivación, las reglas pueden ser de gran importancia debido a Witt, 2012:

1. Reglas que ya soportan sistemas actuales y que deben ser soportadas por nuevos sistemas
2. Objetivos organizacionales que requieran ser controlados y medidos, y
3. Brindan simplicidad en el diseño de nuevos sistemas de información

Por otra parte, la definición formal de reglas de negocio ayuda a la captura del aspecto comportamental de una organización. Dicho aspecto, por lo general documentado en forma de procesos y procedimientos, poseen la desventaja de no contar con algún framework que permita capturar, documentar y ejecutar comportamiento de negocio, labor en la cual el enfoque de reglas de negocio puede ayudar a definir Loshin, 2013.

2.3. Tipos de Reglas

De acuerdo a Witt, 2012, existen dos tipos de reglas:

1. **Reglas Operativas**: Son aquellas que ocurren o no ocurren bajo ciertas circunstancias, con el fin de:
   - Forzar que un dato sea diligenciado completamente
   - Restringir estados en procesos o actividades
   - Restringir la ejecución de ciertas actividades

   Un ejemplo de este tipo de reglas sería: *Todo docente con un Contrato Hora Cátedra debe contar con información de cuenta bancaria*
2. **Reglas Definicionales**: Restringe como definir los elementos estructurales de una organización. Dentro de este grupo se encuentra la definición formal de términos de negocio, periodos de tiempo, categorías, medidas, conceptos complejos y algoritmos.

   Un ejemplo de este tipo de reglas sería: *Toda persona que tenga horas de clases asignadas y posea un Contrato a Término Fijo es denominado Docente de Planta*

2.4. Enfoque de reglas de negocio (Business Rules Approach)

Además de los tipos de reglas expuesto, una regla de negocio puede ser observada desde dos perspectivas, dependiendo sobre el actor a quien aplica (personas / sistemas información) The Business Rules Group, 2000:

- Perspectiva de negocio, la cual son las restricciones que aplican a las personas e interesados del negocio
- Perspectiva de sistemas de información, los cuales son el conjunto de restricciones que aplican sobre los hechos guardados como datos en los sistemas de la empresa.

La perspectiva del negocio es la más administrada, dado que se manifiesta en un conjunto de políticas y decisiones que se ven explícitas en el día a día de la empresa, o en algunos casos, en los procesos documentados por áreas de procesos y calidad.

Por otra parte, aunque los Sistemas de Información que usa una empresa implementan de manera implícita reglas y lógica de negocio, presentan como mayor desventaja que cualquier cambio no es fácil de propagar sin requerir un gran esfuerzo técnico. Esto se ve reflejado en el código legado de las aplicaciones que contiene lógica de negocio, que en caso de que presente cambios no son fáciles de implementar por parte del personal de negocio. Además, el hecho de que las reglas de negocio inmersas en los sistemas de información se mantengan "ocultas" del personal de negocio hace un ambiente peligroso para la toma de decisiones, ya que el personal (e inclusive los desarrolladores del sistema de información) comienzan a realizar suposiciones acerca del sistema desarrollado que pueden llegar a no ser ciertas o ser inexactas von Halle, 2001.

Como mecanismo de mejorar la perspectiva de Sistemas de Información de las reglas de negocio, ha surgido la metodología de enfoque de Reglas de Negocio (Business Rules Approach). El enfoque en reglas de negocio es *"una metodología – y posiblemente tecnología especial – por el cual se captura, gestiona, publica, automatiza y cambian reglas desde una perspectiva estratégica de negocio"* von Halle, 2001. Otra forma de ver este enfoque es la separación efectiva entre la lógica del negocio (es decir en si la definición de las reglas) de su implementación en los sistemas de información Loshin, 2013.

El uso del enfoque, además de permitir un cambio fácil en las reglas de negocio que usan los Sistemas de Información, empoderan a los líderes de negocio para que los sistemas reflejen los mismos cambios que se presentan en el negocio, a un ritmo constante y casi inmediato. El hecho de adaptar cambios de forma rápida y concisa es una característica invaluable en el día de hoy para las empresas, ya que se encuentran en un proceso constante de asumir nuevo conocimiento, corregir conocimiento anterior, aplicar dicho conocimiento en el comportamiento diario y evaluar sus resultados, en un proceso conocido como aprendizaj evon Halle, 2001.

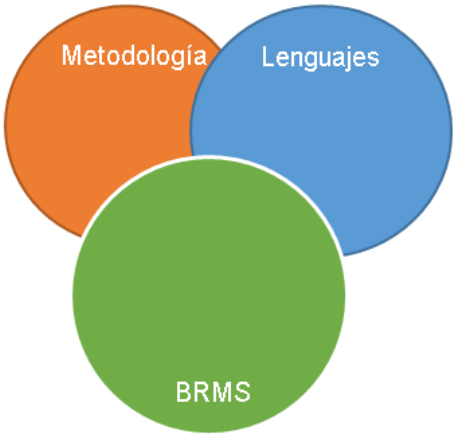
El resultado de aplicar el enfoque es la creación de un sistema de reglas de negocio. Dicho sistema hace posible la separación lógica (y posiblemente física) de las reglas de negocio de otros aspectos del sistema, haciendo posible que sean compartidas a través de almacenes de datos, interfaces de usuario y aplicaciones von Halle, 2001. El manejo de las reglas de negocio a través de un sistema habilita a una organización la externalización y manejo del conocimiento, sobre el cual los demás sistemas vinculados serán diseñados y administrados.

2.5. Implementación de un Enfoque de Reglas de Negocio

Para implementar un enfoque de Reglas de Negocio en una organización (y puntualmente en los Sistemas de Información desarrollados de una compañía), se debe tener en cuenta los siguientes tres componentes (ver Figura 2.1) Boyer y Mili, 2011:

1. Una metodología para la administración de reglas de negocio, esto es, *"recolectar, validar, evaluar, publicar y evolucionar las reglas de negocio"*
2. Uno o más lenguajes formales para expresar reglas de negocio en diferentes etapas de su ciclo de vida
3. Un Sistema de Administración de Reglas de Negocio (BRMS por sus siglas en inglés)

Figura 2.1: Componentes de un Enfoque de Reglas de Negocio



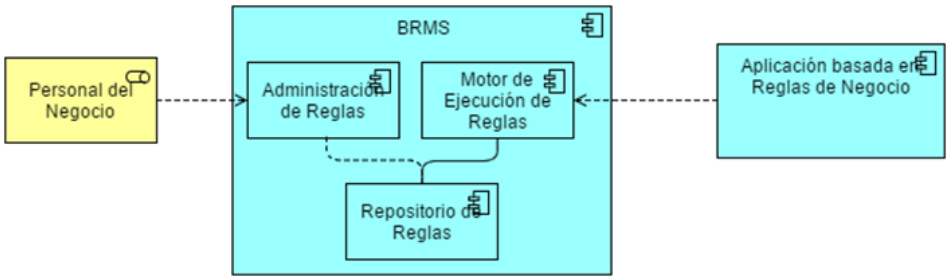
Fuente: Boyer y Mili, 2011

Respecto a los tres componentes mencionados, el BRMS es el artefacto que permite materializar tanto la metodología como los lenguajes necesarios para definir reglas de negocio. Por una parte, el BRMS establece un repositorio compartido para artefactos de reglas y su administración, de esta forma definiendo una forma estructurada y organizada de trabajo (es decir, una metodología). Los artefactos de dicho repositorio son definidos utilizando una o mas lenguajes de definición de reglas, las cuales son fácilmente administrables a través de una interfaz gráfica. Debido a la importancia que tiene el BRMS para el enfoque de reglas de negocio, es importante identificar sus componentes principales y comprender su funcionamiento.

2.5.1. El sistema de Administración de Reglas de Negocio - BRMS

El BRMS (Business Rules Management System) es un componente que permite realizar dos funciones básicas: la administración de las reglas de negocio y la ejecución de las mismas. La administración es realizada mediante un componente gráfico que permite que personas del negocio puedan realizar cambios sobre el repositorio de reglas de negocio. Por otra parte, el motor de ejecución de reglas actúa bajo demanda, de acuerdo a las solicitudes realizadas por otras aplicaciones de negocio, y tiene acceso de sólo lectura a las reglas de negocio del repositorio (Figura 2.2).

Figura 2.2: Arquitectura de un BRMS



Fuente: Boyer y Mili, 2011

2.6. Relación con BPM

2.6.1. Introducción

Desde sus inicios, el enfoque basado en reglas de negocio ha tenido una íntima relación con la metodología de administración de procesos de negocio (*BPM - Business Process Management*). Lo anterior dado que las empresas, en su afán de obtener beneficios para los Stakeholders u obtener ganancias White y Miers, 2008, pg. 19, se han enfocado en mejorar sus procesos operacionales. Dichos procesos poseen como entradas *"objetivos, estrategias y reglas (o regulaciones) de la organización"* White y Miers, 2008, pg. 20, los cuales son procesados por un conjunto de tareas (o acciones) y actores que la ejecutan McKinty y Mottier, 2016, pg. 1. Es así como surge BPM, el cual es un enfoque y una tecnología para *"hacer los procesos mas eficientes, flexibles y dependienes, y para reducir costos"*McKinty y Mottier, 2016, pg. 4, el cual establece una cultura dirigida a la mejora contínua de los procesos alineados con las estrategias organizacionales Harmom y Tregear, 2016, pg. 2.

Uno de los principales beneficios de BPM a nivel organizacional es que permite la generación de modelos de procesos de negocio que faciliten la comunicación con otros miembros de la organización White y Miers, 2008, pg. 21. Fomentar el uso de lenguaje de dominio común entre los miembros de una organización da origen al surgimiento de identificación de mejoras, construidas colectivamente.

Otros beneficios que trae la aplicación de BPM son Rosing, Scheer y Scheel, 2015, pg. 508:

- Mejora la adaptabilidad y tiempo de respuesta de una organización frente a los cambios
- Estimula la innovación en productos, servicios, procesos y modelos de negocio
- Reduce los costos y mejora la productividad de los recursos
- Mejora la calidad de los productos generados como resultado de los procesos
- Mejora la coordinación entre áreas de una empresa, y la sincronización entre las actividades que ejecutan.

2.6.2. Conceptos Clave

Para hablar de BPM, es necesario definir los conceptos que se manejan en el ámbito de esta disciplina. Lamentablemente, hay diversidad de frameworks y variedades de BPM en el mercado que hacen que algunos términos tengan definiciones ambiguas que impiden tener una visión unificada Silver, 2006, pg. 27. Para ello, se acudirá a las definiciones dadas por Rosing y col., 2015, pg. 101, y las cuales se resumen en la tabla 2.1.

Tabla 2.1: Conceptos Clave BPM

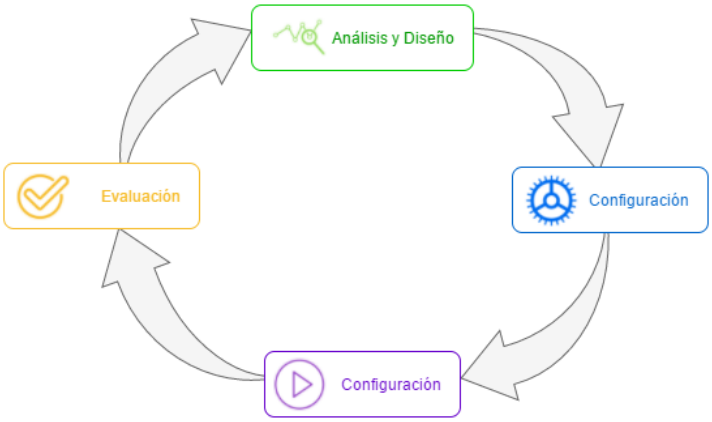
Concepto	Definición
Proceso de Negocio	Conjunto estructurado de actividades o tareas con comportamientos lógicos que produce un servicio o producto específico
Paso de Proceso	Un conjunto conceptual de comportamientos delimitados por el alcance de un proceso que, cada vez que es ejecutado, lleva a un cambio simple de entradas a una sola salida. Cada paso de proceso es una unidad de trabajo normalmente realizada dentro de los límites de un conjunto de reglas por uno o mas actores en un rol, que modifica el estado de uno o mas recursos u objetos empresariales.
Actividad de Proceso	Una parte de un sistema físico que especifica como completar un cambio o la interacción con otros actores a partir del conocimiento, juicio o experiencia.
Evento	Un cambio de estado que reconoce la ejecución o terminación del procesamiento
Compuerta	Determina la bifurcación o unión de rutas de ejecución de un proceso, dependiendo de las condiciones expresadas
Regla de Proceso	Una expresión que define o restringe algún aspecto del trabajo y siempre resuelve a verdadero o falso
Medida de Proceso (Indicadores de Proceso)	Base sobre la cual una organizacion evalúa si un proceso o actividad se desempeña como es esperado.

Fuente: Rosing, Scheer y Scheel, 2015, pg. 102-103

2.6.3. Ciclo de Vida

El trabajo llevado a cabo para aplicar la disciplina BPM en la organización posee distintos enfoques. Prácticamente, cada autor que ha escrito de BPM ofrece un enfoque distinto de ciclo de vida, pero que aunque aplican diferentes términos, se logra identificar coincidencias conceptuales entre ellos, tal como quedó en evidencia en Galvis Lista y González Zabala, 2014. De acuerdo a lo anterior, se logra identificar un conjunto común de fases, las cuales se pueden observar en la Figura 2.3.

Figura 2.3: Ciclo Vida BPM



Fuente: Galvis Lista y González Zabala, 2014, pg. 40

- *Análisis y Diseño*: Define el *As-Is* (o estado actual) y el *To-Be* (o estado ideal) de la organización, definiendo los objetivos de cada proceso en términos de eficiencia y efectividad. De igual manera, se definen acuerdos de niveles de servicio y detalles del proceso como actores, notificaciones y escalamientos Rademakers y Liempd, 2010, pg. 6.
- *Configuración*: Consiste en configurar el proceso definido en el Análisis sobre el entorno de ejecucion BPM.
- *Ejecución*: Consiste en realizar seguimiento y monitoreo de las instancias de los procesos que se ejecutan en el entorno de ejecución BPM.
- *Evaluación*: Se identifica fortalezas y oportunidades de mejora en el (los) proceso(s).

2.6.4. Modelado con BPMN

El resultado de aplicar BPM en su fase de Análisis y Diseño es obtener por cada proceso uno o mas modelos de proceso, que pueden llegar a utilizar diferentes notaciones gráficas dependiendo del tipo de framework con el cual se diseñaron los procesos. La notación mas empleada en la actualidad es BPMN (Business Process Model and Notation), notación estandarizada por la organización OMG (*The Object Management Group*), y cuyo propósito es

suministrar una notación que sea fácilmente comprendida por todos los usuarios de negocio, y un puente estandarizado que cierre las brechas entre el diseño e implementación de un proceso The Object Management Group Inc., 2010, pg. 1.

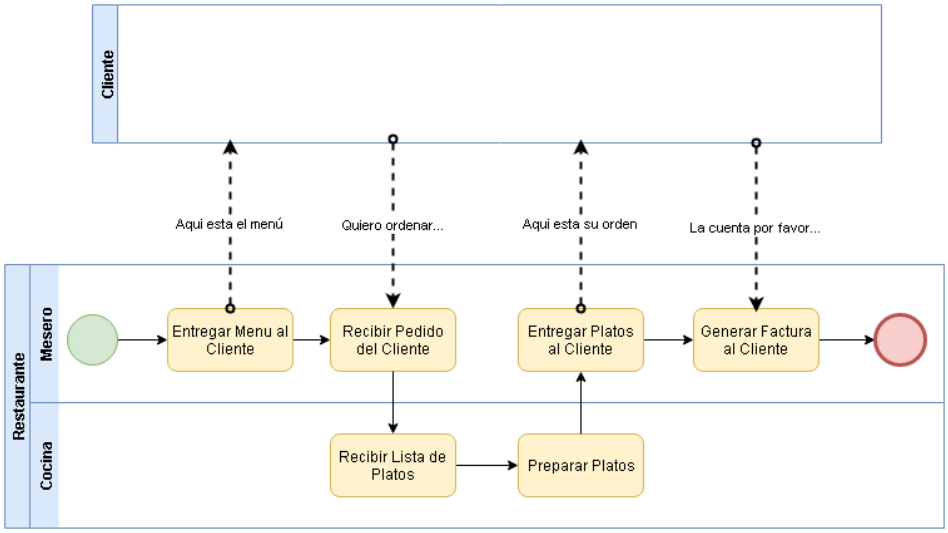
La idea inicial de la notación era suministrar una forma de notación gráfica del flujo de un proceso que permitiera una trancisión rápida a la generación de su especificación respectiva en términos de un lenguaje de ejecución XML The Object Management Group Inc., 2010, pg. 21. Dado que uno de los fines últimos de BPM es suministrar una vía para la automatización de procesos de negocio, el problema que se presentaba durante la implementación de los proyectos de BPM era que la especificación ejecutable de los procesos de negocio se realizaba en un formato que no era comprensible por analistas de negocio. Por tanto, BPMN surge como un enfoque que suministra varios diagramas, que además de brindar una notación gráfica amigable, facilita el mapeo a los lenguajes de ejecución que manejan los motores de ejecución BPM The Object Management Group Inc., 2010, pg. 21.

El lenguaje puede ser utilizado para los siguientes usos The Object Management Group Inc., 2010, pg. 22:

- *Creación de Procesos (Orquestación)*, tanto para uso privado al interior de las organizaciones, como para uso público y externo de los usuarios.
    - Cuando se trata de procesos *privados*, la secuencia de acciones/actividades ocurre dentro de un solo *Pool*, representando un solo proceso
    - Cuando se diseñan o ejecutan procesos *públicos*, estos suceden entre dos *pools*, y dado que cada Pool representa un prceso o participante diferente, la única forma de comunicación entre ellos es a través del intercambio de mensajes.

      Al diseñar este tipo de procesos, por lo general, se pueden llegar a generar diferentes *Puntos de Vista*, los cuales pueden hacer ver a un participante como invisibles las actividades ejecutadas en otro Pool, ya que se trata de un actor externo del proceso (ver Figura 2.4).
- *Colaboraciones*, que ilustran la interacción entre dos o mas entidades de negocio, a través del intercambio de mensajes.
- *Coreografías*, las cuales capturan los flujos de mensajes explícitos que existen entre procesos. Para ello, BPMN ha creado una notación gráfica correspondiente.
- *Conversaciones*, las cuales representan la relación lógica de los intercambios de mensajes que ocurren entre procesos, y que involucran objetos de negocio.

Figura 2.4: Ejemplo de un punto de vista en BPMN



Fuente: Autor

2.6.5. Actividades y tipos de actividades

En un Pool se encuentra una serie de actividades, las cuales pueden ser atómicas o compuestas The Object Management Group Inc., 2010, pg. 32. Una actividad atómica (o también denominada tarea) es aquella que no puede ser descompuesta en mas pasos. En cambio una actividad compuesta (o también denominada subproceso) está compuesta por mas actividades atómicas o compuestas.

A su vez, las *tareas* se clasifican en White y Miers, 2008, pg. 68:

- *Ninguna*, las cuales son tareas genéricas sin ningún tipo de clasificación
- *Manual*, las cuales son tareas no automáticas realizadas por un humano
- *Recepción*, las cuales se encuentran en espera de un mensaje que llega de un participante externo. Una vez se recibe el mensaje externo, la tarea se completa.
- *Script*, las cuales ejecutan un script definido por el diseñador del modelo
- *Envío*, las cuales envían un mensaje a un participante externo
- *Servicio*, las cuales se vinculan a algún tipo de servicio, el cual puede ser un WebService o una aplicación.
- *Usuario*, en la cual una persona realiza una acción con ayuda de una aplicación (por lo general en forma de recepción de mensajes en una Bandeja de Entrada de tareas pendientes).

2.6.6. Automatización de procesos y su relación con las Reglas de Negocio

Los procesos que pueden ser candidatas a la automatización son aquellos que posean alguna de las siguientes características McKinty y Mottier, 2016, pg. 5:

- Uso Frecuente
- Contiene tareas de aprobación
- Uso concurrente
- Contiene tareas sin intervención humana que de hecho requieren supervisión
- Fuerte necesidad de consistencia y medida
- El proceso es crítico

Al seleccionar un proceso para ser automatizado, es importante definir aquellas tareas que son suceptibles de ser automatizadas. Tal como se definió previamente en los tipos de tareas, las definiciones de automatización se pueden dar a nivel de script o a nivel de Servicios (por medio del consumo de Web Services). Una vez definida las tareas a automatizar, se debe proceder a crear las aplicaciones que cumplan con dicho fin.

Entre las tareas que se encuentran en los flujos de procesos para automatizar nos podemos encontrar con aquellas que generan una decisión de acuerdo a una serie de reglas de negocio que se aplican sobre los datos del flujo del proceso. Por lo general dichas tareas quedan consignadas como manuales o de usuario, ya que lamentablemente los criterios de decisión son puestos a consideración del participante en el proceso. Sin embargo, y de acuerdo al enfoque de reglas de negocio explicado previamente, el conocimiento y los criterios utilizados por el personal de negocio podría ser fácilmente extraído en un repositorio de reglas, el cual podría exponer su funcionalidad a través de un Web Service. De este modo, dónde antes las tareas se encontraban como *Manuales o de Usuario* podrían volverse automáticas (o tareas de *Servicio*) a través del consumo del Web Service que genere una decisión a partir de la aplicacion de las reglas centralizadas en un repositorio.

2.7. Diferencias con el desarrollo tradicional de aplicaciones

El uso de un enfoque basado en reglas de negocio implica como principal actor el sistema de administración de reglas de negocio (BRMS), desde el cual se consultan y consumen las reglas administradas por los líderes del negocio. Esto implica cambios sustanciales respecto al desarrollo tradicional de Software, en dónde las reglas eran por lo general escritas como parte del código de la aplicación, y utilizadas como pre y post condiciones a procesos propios de la aplicación Cemus, Cerny y Donahoo, 2015. Las diferencias respecto al desarrollo tradicional radica en Cemus y col., 2015:

- *El código*, el cual se diferencia debido a que existen partes del código cuya responsabilidad serán delegadas al BRMS. Esto implica que en algunas funciones del código no existirán referencias a condicionales o ciclos, en especial aquellas relacionadas con validación del flujo de datos de la aplicación.

- *Despliegue*, el cual respecto a una aplicación tradicional se divide en: (a) las reglas de negocio administradas por un BRMS y (b) una infraestructura computacional encargada de las demás funciones de la aplicación (interfaz gráfica, capa de lógica, capa de persistencia, etc). Ambos componentes "son empaquetados y desplegados por separado, y frecuentemente de manera asíncrona" Boyer y Mili, 2011.
- *Tiempo de ejecución*, en dónde los resultados de una aplicación tradicional vs una aplicación con enfoque de reglas de negocio no deberían presentar diferencias en sus resultados durante el tiempo de ejecución, ya que están utilizando como origen las mismas reglas de negocio.
- *Mantenimiento*, el cual es más rápido y fácil de administrar por parte del negocio, dado que cualquier cambio relacionado con las reglas del negocio podrán ser administrados por el BRMS. Por otra parte, cualquier introducción de nuevas reglas de negocio podrá ser implementado sin problemas y propagado a todas las aplicaciones relacionadas con el BRMS, por tratarse de componentes independientes en su funcionamiento.
- *Reuso de reglas*, en especial en las capas del desarrollo correspondientes a Interfaz Gráfica, Capa de Servicios y Capa de Persistencia, dónde en la mayoría de desarrollos tradicionales se presenta una repetición de reglas que presentan dificultades al momento de realizar ajustes Cemus y col., 2015.

Además de las características descritas respecto al mantenimiento, existen otros factores que hacen que una aplicación con enfoque en reglas de negocio sea más fácil de administrar Boyer y Mili, 2011:

- Comprensión de las reglas por parte del negocio, evitando el *Conocimiento Tribal*, el cual consiste que la propiedad y comprensión de procedimientos y reglas sin documentar se encuentran solo en propiedad de algunas personas del negocio Ary, 2013, pg. 3.
- Despliegue independiente
- Ejecución por separado de la aplicación (a través del BRMS)

CAPÍTULO 3
Pruebas de Software y Automatización

3.1. Contexto: Pruebas de Software

La fase de pruebas del ciclo de vida del Software es definida como ”un proceso o una serie de procesos diseñados para asegurar que el código cumple para lo que está diseñado y no haga nada de forma involuntario” Myers y Sandler, 2004. Para lograr realizar la verificación del código contra su diseño (y por tanto, contra los requisitos del cliente), se toman datos de ejemplo de entrada, y se plantean salidas esperadas de acuerdo a los requerimientos. Para realizar una fase de pruebas correctamente, se recomienda responder las siguientes preguntas, antes inclusive de contratar al equipo de pruebas Black, 2009:

- ¿Que podría probar?
- ¿Qué debería probar?
- ¿Qué puedo probar?

En lo que debería probar, debería priorizarse aquellas áreas en las cuales no se tiene cobertura de pruebas. Por tanto, al ejecutar la fase de Pruebas, es mejor evitar la creencia popular de evaluar distintos tipos de entrada a las cuales se generarán distintos tipos de salida, alcanzando una permutación entre entradas lo suficientemente completa para garantizar la calidad de las salidas del Software. Crear casos de prueba para todas las posibles entradas y salidas es impráctico, y requeriría de muchos recursos humanos para ser económicamente factible Myers y Sandler, 2004.

Un enfoque más racional estaría enfocado en probar el software con el fin de buscar errores durante su ejecución. La anterior afirmación podría ir en contravía con la creencia común de que las pruebas se ejecutan para verificar que un programa cumple con los requisitos declarados por el cliente, lo cual en parte es cierto, pero para que el proceso de pruebas agregue valor al proceso es importante que demuestre que el software bajo prueba (o Software Under Test SUT) posee cualidades de confiabilidad, y para lograr demostrar ello es necesario detectar y remover errores Myers y Sandler, 2004.

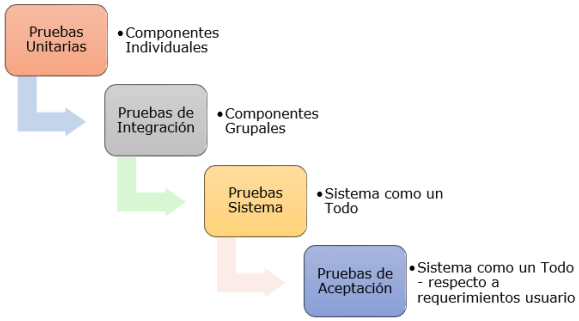
En cuanto a lo que se debería probar, el enfoque está en identificar aquellos errores o bugs que afecten la experiencia del cliente respecto a la calidad del producto. Los enfoques de pruebas deberían enfocarse a encontrar los defectos críticos que limitan la habilidad de

los usuarios de llevar a cabo su trabajo usando el software desarrollado Black, 2009. Respecto a lo que se puede probar, es revisar el recurso humano, financiero y de tiempo con el que se cuenta para realizar pruebas, y verificar que tanto de lo que "se debería probar" se puede realizar con los recursos disponibles.

3.2. Niveles de Pruebas

La ejecución de pruebas posee 4 niveles de pruebas, los cuales dependen en gran medida de la fase en la cual se encuentra en ejecución el proyecto de software Black, 2009. Los niveles se encuentran ilustrados en la Figura 3.1.

Figura 3.1: Niveles de Pruebas



Fuente: Autor

- *Pruebas Unitarias*, la cual se define como la unidad más pequeña de software que se puede probar. Las unidades para realizar pruebas a este nivel son: procedimientos y funciones, clases y sus métodos Burnstein, 2010

- *Pruebas de Integración*, cuyo propósito es detectar defectos que pueden suceder en las interfaces de las unidades de software (procedimientos, funciones, clases) una vez se ponen en funcionamiento con otras unidades; y ensamblar dichas unidades en subsistemas funcionales que puedan ser posteriormente probados en las pruebas de sistema Burnstein, 2010

- *Pruebas de Sistema*, cuyo objetivo es verificar que el sistema se comporta de acuerdo a los requerimientos. Evalúa tanto funcionalidad como atributos de calidad tales como confiabilidad, usabilidad, rendimiento, y seguridad. Esta fase es especialmente útil para detectar defectos con interfaces externas de Hardware y Software Burnstein, 2010.

- *Pruebas de Aceptación*, las cuales son realizadas por los usuarios finales en conjunto con los testers, con el fin de verificar que sus requerimientos y expectativas hayan sido cumplidas en el producto final.

3.3. Automatización de Pruebas

La automatización de pruebas puede ser definida como *"el uso de software para controlar la ejecución de pruebas, la comparación de las salidas actuales con salidas predecidas, la configuración de las precondiciones, y otras funciones de control y reportes"* Amaricai y Constantinescu, 2014, pg. 1. De ahí que las pruebas automatizadas generan como resultado scripts que a su vez generan reportes sobre la efectividad de las pruebas ejecutadas sobre el Software Bajo Prueba (Software Under Test SUT).

El uso de la automatización de pruebas ayuda en gran medida al personal de Calidad. Lo anterior dado que por lo general las organizaciones asumen que cuando un software pasa a la fase de pruebas, en dicha fase serán detectados todos los bugs posibles, lo cual en la práctica es imposible, dado la ausencia de los recursos necesarios para hacerlo. Por tanto, con el uso de tests automáticos los testers podrán dedicar más tiempo a escribir más casos de prueba para cubrir más aspectos del sistema, y delegar todo el tema de validación y evaluación a los scripts de prueba Yu y Patil, 2007. De igual manera, el uso de scripts de prueba mejora la calidad de las pruebas, originado por la mínima intervención humana que requieren durante su ejecución Gojare, Joshi y Gaigaware, 2015.

Entre otras ventajas de la automatización de pruebas se encuentra Joy y Singh, 2015, pg. 1:

- Optimización de la velocidad, eficiencia y calidad del SUT
- Costo reducido en la ejecución de pruebas manuales
- Mayor retorno a la inversión (ROI)

Aunque el uso de la automatización de pruebas puede traer venajas a nivel de tiempo y recursos financieros, su aplicación puede llegar a requerir cambios en las habilidades del equipo de trabajo que implementa las pruebas. Lo anterior debido a que el ámbito de implementación de la automatización de pruebas puede llegar a situarse entre pruebas y desarrollo Amaricai y Constantinescu, 2014, pg. 1. Lo anterior dado que la automatización requiere de scripts para que puedan ser ejecutados y reproducidos varias veces. Por otra parte, es necesario por parte de quien crea los scripts tener suficiente conocimiento en pruebas y de negocio para generar automatizaciones correctas a los requerimientos del cliente.

De acuerdo a ello, las pruebas automatizadas tienen el mismo carácter y estructura que un proyecto normal de software. De ahí que requiera características tales como manejo externo de archivos, interacción con interfaz de usuario y manejo de plantillas de casos de pruebas. Para la administración de cada uno de los artefactos que participan en la autmatización de pruebas, surge como respuesta la creación de Frameworks de Automatización.

3.3.1. Frameworks de Automatización

Un framework de automatización es definido como un *"conjunto de conceptos, procesos, procedimientos y entornos abstractos en los cuales los tests automáticos son diseñados, creados e implementados"* Amaricai y Constantinescu, 2014, pg. 2. Además, un framework proporciona

"guias, estándares de código, (...), prácticas, jerarquías de código, modularidad y mecanismos de reporte" Joy y Singh, 2015, pg. 1 a los ingenieros y personas encargadas de la implementación de la automatización de pruebas. La funcionalidad central de estos frameworks va enfocada a generar logs y reportes de la ejecución de los scripts de automatización, además de permitir extender su funcionalidad agregando librerías.

El uso de un framework de automatización de pruebas posee las siguientes ventajas Joy y Singh, 2015, pg. 1:

- Reusabilidad del código
- Costo mínimo de mantenimiento
- Intervención manual mínima
- Facilidad en la generación de reportes

Los frameworks se diseñaron teniendo en cuenta los aspectos que se requieren para probar una aplicación de forma automática, a saber Amaricai y Constantinescu, 2014, pg. 2:

- *Caso de prueba o flujo de prueba*
- *Script de prueba*
- *Datos de prueba*
- *Localizadores*, que son todos aquellos identificadores que permiten ubicar en la interfaz gráfica elementos como botones, cuadros de texto, etc.

De acuerdo a la forma en la cual se implementan cada uno de los aspectos mencionados, se pueden detectar los siguientes tipos de frameworks que pueden ser implementados Joy y Singh, 2015, pg 1:

- *Orientados a Datos*, en los cuales la lógica de los scripts de prueba son separados de los datos de prueba, los cuales son almacenados en bases de datos externas tales como archivos XML, CSV, Excel, bases datos ODBC, etc.
- *Orientados por palabras claves*, en los cuales las acciones que suceden en las pruebas son almacenadas en archivos externos en forma de "palabras clave", en dónde el conjunto de palabras clave indica al script que debe hacer.
- *Híbridas*, en las cuales datos y acciones son almacenados en archivos externos. Además, los archivos también contienen identificadores de localizadores, formato de mostrar los mensajes de errores, escenario de transacciones, etc.

Pero, ¿como saber cual de los tipos de frameworks implementar? Dicha decisión dependerá en gran medida de los casos de prueba que se deben implementar, al igual que los requerimientos que el SUT exija para su prueba Joy y Singh, 2015, pg 1. Por otra parte, además de escoger un tipo de framework, es importante escoger un enfoque de diseño que permita implementar el tipo de framework escogido.

3.3.2. Enfoques de Diseño de Frameworks de Automatización

Page Object

El enfoque de diseño más popular es el *Page Object*, el cual se encuentra dentro de la clasificación de orientados por palabras clave. Dicha implementación permite que las pruebas puedan acceder a funcionalidades del SUT a través de funciones sencillas de una clase, y dichos métodos contienen a su vez las acciones que se deben realizar sobre la interfaz gráfica para logar el objetivo de la función/método invocado Amaricai y Constantinescu, 2014, pg. 3.

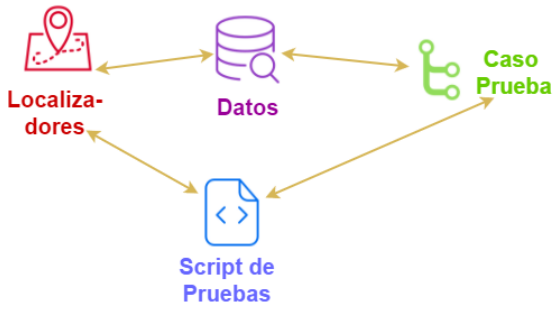
Page Object posee ciertas ventajas y desventajas en su implementación. Dentro de sus ventajas se encuentra que cualquier cambio en el SUT solo requerirá modificación de la clase que implementa el diseño y no de los tests. Por otra parte, presenta como desventaja que requiere de testers que tengan de cierto nivel de programación, además de que su tiempo de mantenimiento incrementa de manera lineal cuando existen más de dos aplicaciones con un promedio de 10 – 20 páginas Amaricai y Constantinescu, 2014, pg. 3. A pesar de sus desventajas, Page Object al día de hoy se ha convertido en la implementación mas extendida en varios frameworks reconocidos.

Otros Enfoques de Diseño

Frente a las desventajas presentadas por Page Object, otros autores han intentado implementar otros enfoques de diseño, dirigidos en gran medida a la implementación de un tipo de framework híbrido. Entre las distintas propuestas es de resaltar las siguientes:

Enfoque diseño Amaricai/Constantinescu, el cual parte del problema de mantenibilidad de los scripts una vez el volumen de Page Objects comienzan a incrementar, y la dificultad de tener personal de pruebas con la habilidad necesaria para realizar mantenimiento a los scripts. Para ello, proponen externalizar los aspectos de datos, localizadores y flujos de ejecución del test en archivos, de tal manera que el mayor esfuerzo requerido sea a nivel de integración de dichos archivos, y un esfuerzo mínimo durante el mantenimiento Amaricai y Constantinescu, 2014, pg. 156. El esquema de funcionamiento se describe en la Figura 3.2

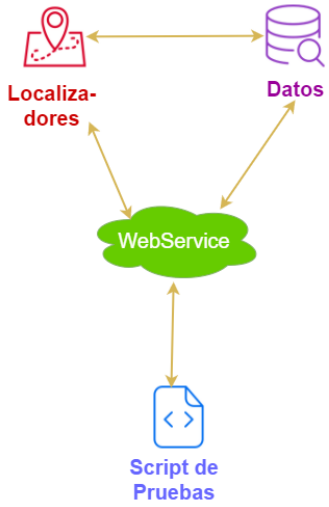
Figura 3.2: Estructura Enfoque Diseño Amaricai/Constantinescu



Fuente: Amaricai y Constantinescu, 2014, pg. 157

Diseño de Framework Genérico, creado por Joby Joy y Devendra Pratap, parte del problema de que existen distintas fuentes de datos sobre las cuales un script de pruebas puede ejecutarse (XML, Excel, JSON, etc), dependiendo de la implementación de Framework escogida. Por tanto, proponen un modelo de diseño que parte de la existencia de otros frameworks de automatización que reciben/generan distintos formatos de datos de salida, para los cuales emplea un Web Service que realiza la conversión entre los diferentes tipos de archivos. El esquema de funcionamiento se describe en la Figura 3.3.

Figura 3.3: Estructura Enfoque Diseño Framework Genérico



Fuente: Joy y Singh, 2015, pg. 208

3.3.3. Mejores Prácticas en la Automatización de Pruebas

Con el fin de garantizar el éxito en los proyectos de automatización de pruebas, el ISTQB (International Software Testing Qualifications Board) ha propuesto una serie de buenas prácticas en el marco de la certificación en Automatización de Pruebas. Dichas prácticas son enfocadas como criterios de éxito, además de una lista de lo que se debe y no se debe hacer en este tipo de proyectos.

Como buenas prácticas se propone International Software Testing Qualifications Board - ISTQB, 2016, pg. 13:

1. *Tener establecida una Arquitectura de Automatización de Pruebas (TAA - Test Automation Architecture)*, teniendo en cuenta que la automatización de pruebas es un proyecto de Software, y debe ser tratado como tal.

2. *El SUT (Software Under Test) debe ser desarrollado para soportar pruebas automáticas*, en dónde el SUT debe desacoplar la interacción con la interfaz gráfica de los datos y su apariencia visual. Esto se puede traducir en la creación de mecanismos para identificar de manera única cada elemento visual presente en el Software Under Test.

3. *Identificar las partes del SUT que son sensibles de ser probadas, e iniciar el proyecto por dichas partes*. Realizar esta acción permitirá mostrar resultados rápidos en la ejecución del proyecto de automatización de pruebas, mostrando la facilidad de implementación de los scripts de prueba.

4. *Establecer una estrategia de Automatización de Pruebas*. Según el Instituto Internacional para Pruebas de Software (International Software Test Institute), una estrategia de pruebas debería definir International Software Test Institute, 2016:

    - Las secciones del software que requieren auntomatización de pruebas
    - Cómo la tarea se llevará a cabo
    - Como y dónde los scripts de pruebas tendrán mantenimiento
    - Cómo el proyecto se beneficiará de esta actividad
    - Cuanto será el ahorro en costos

5. *Seleccionar o Crear un Framework de Automatización de Pruebas*. Al respecto de este punto, ISTQB recomienda que el Framework que se escoja en conjunto con la arqutectura de automatización de pruebas debe cumplir con los siguientes requisitos:

    - Tener facilidad para generar reportes que muestren la calidad del SUT
    - Permitir identificar la causa de la falla en la ejecución de las pruebas, ya sea que se trate del SUT o de fallas en los scripts de pruebas
    - Tener definido claramente el entorno de pruebas requerido para la correcta ejecución del Framework
    - Facilitar la documentación de los casos de pruebas
    - Facilitar la trazabilidad de los pasos de los casos de pruebas
    - Permitir un fácil mantenimiento de los scripts

Por otra parte, ISTQB recomienda como lo que no se debe hacer en un proyecto de automatización de pruebas:

1. Crear scripts con alta dependencia a la interfaz gráfica, es decir, que no dependan de los cambios en la apariencia visual del SUT

2. Scripts con alta dependencia de ciertos tipos de datos del SUT o a valores fijos

3. Scripts que dependan del contexto de la prueba o de su ambiente de ejecución (ej: ubicación geográfica del Sistema Operativo en dónde se ejecute la automatización de pruebas).

La verificación de algunas de las buenas prácticas mencionadas se puede realizar inclusive antes de la ejecución del proyecto, aunque es de aclarar que es muy extraño ver iniciativas que cumplan con todos los criterios International Software Testing Qualifications Board - ISTQB, 2016, pg. 14.

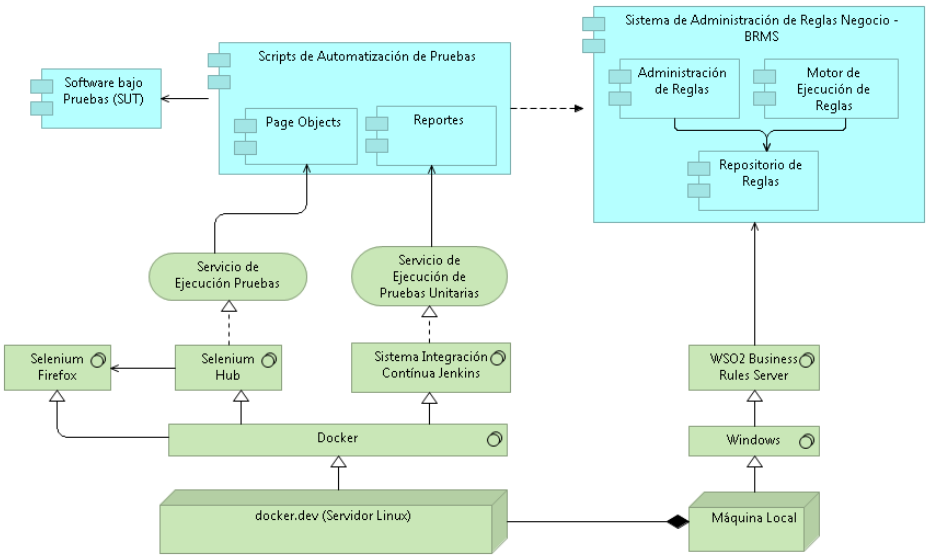
CAPÍTULO 4

Integración BRMS - Framework Automatización de Pruebas

4.1. Descripción de la solución

Con el fin de realizar la integración entre reglas de negocio y un framework de Automatización de pruebas y una vez realizada una exploración preliminar sobre las tecnologías disponibles, se plantea la arquitectura descrita en la Figura 4.1.

Figura 4.1: Arquitectura General de la Solución Propuesta



Fuente: Autor

4.1.1. Selección del Software Bajo Pruebas (SUT)

El primer paso para la aplicación de esta solución es escoger una aplicación o Software Bajo Pruebas (Software Under Test - SUT) al cual se desea automatizar sus pruebas. Como se ha identificado en otros escenarios, la automatización de pruebas no es aplicable a todo tipo de Software y Componentes, y en particular para el presente enfoque adoptado se identifica que no es posible automatizar pruebas de software cuyas reglas de negocio no se encuentren definidas, o de los cuales se necesite validar datos que no sean dependientes de la información que se muestre en las vistas.

Por tanto, las condiciones de selección del Software deberá cumplir con las siguientes condiciones:

- Deberá ser regido por una o mas reglas de negocio
- Las validaciones o pruebas requeridas sobre su funcionalidad parten de la información que se puede ver en la vista o vistas del componente/aplicación
- Deberá ser una aplicación Web

4.1.2. Selección del Sistema de Administración de Reglas de Negocio - BRMS

Hoy en día existen varios productos Open Source y comerciales que implementan el enfoque de Reglas de Negocio explicado previamente. Dado la variedad de alternativas disponibles y con el fin de evitar problemas de Licenciamiento, para evaluar el BRMS idóneo para este proyecto de grado se decidió comparar las diferentes aplicaciones Open Source referentes en el mercado. Al respecto, se encontraron tres productos que son destacados en aspectos técnicos y funcionales, los cuales son descritos a continuación.

OpenRules

OpenRules[1] es definido como un Motor de Administración de Reglas de Negocio y Decisiones (BRDMS[2]). Brinda una interfaz amigable para analistas de negocio para la creación, modificación y mantenimiento de reglas de negocio, además de facilitar tomar como fuente de reglas diversos orígenes de datos (Hoja de cálculo, Google Spreadsheets, entre otras).

Entre sus principales ventajas se encuentran:

- Utiliza MS Excel como herramienta para la construcción, validación y ejecución de reglas de negocio. Lo anterior facilita la participación de las personas de negocio en la consutrcción de reglas.
- Establece una separación clara entre componentes administrados por personal de TI y aquellos que son del ámbito de los analistas de negocio. De esta manera, los especialistas de TI pueden integrar decisiones creadas por los analistas de negocio con el modelo de objetos disponible sin producir mayores cambios en la lógica de negocio implementada en las aplicaciones.

[1] http://openrules.com/

[2] Business Rules and Decision Management System

- Posiblidad de implementar formularios Web utilizando Excel como herramienta de modelado

Esper

Esper[3] es un componente para procesamiento de eventos complejos (*CEP* - Complex Event Processing). Permite el procesamiento de grandes volúmenes de mensajes o eventos, sin importar su naturaleza o si son históricos. Esper actua como un motor de eventos en tiempo real que ejecuta acciones cuando ciertas condiciones occurren en el stream de eventos. Para realizar el proceso de *matching*, emplea un DSL[4] denominado EPL - Event Processing Language, el cual permite registrar consultas que señalan al Motor sobre que eventos realizar la escucha. Luego que un evento se cumple, Esper hace un llamado a una clase *Listener* , la cual es una clase POJO (Plain Old Java Object) que almacena la información del evento EsperTech Inc, 2016.

Esper actua como una base de datos en memoria, pero en vez de almacenar datos y ejecutar consultas sobre la misma, el motor permite a las aplicaciones almacenar consultas y pasar a través de ellas datos. La respuesta de Esper es en tiempo real cuando las condiciones de las consultas se cumplen EsperTech Inc, 2016.

Aunque su enfoque principal sea en procesamiento de eventos, su ámbito de uso aplica como Motor de Reglas de Negocio debido a que permite la definición de condiciones que, cuando se cumplen, señalan acciones personalizdas que se pueden ejecutar sobre los objetos de negocio.

Entre las ventajas de uso de este motor se encuentran:

- Permite la detección en tiempo real de eventos sobre una cantidad masiva de datos (o mensajes).
- Esper (y su variación para .NET denominada *NEsper*) son componentes que pueden ser incrustados en cualquier aplicación Java (o .NET). Por tanto, son compatibles con cualquier servidor de aplicaciones disponibles en el mercado EsperTech Inc, 2016.
- Permite una representación de los eventos en una representación de objetos de dominio debido a que Esper sooport atodos los aspectos del diseño orientado a objetos al igual que una identificación dinámica del tipo de dato en tiempo de ejecución (denominado *Dynamic Typing*).
- Ofrece una API (*Statement Object Model API*) que permite la detección de cualquier tipo de patrón de evento, además de construir, manipular o interrogar sobre expresiones EPL.

Drools

Drools es un Motor de Reglas de Negocio (BRMS) creado por Jboss Inc, el cual tiene como fin realizar la separación de las reglas de negocio incrustadas en el código fuente de

[3]http://www.espertech.com/esper/

[4]Domain Specific Language - Lenguaje de Dominio Específico

las aplicaciones a través de la definición externa de archivos de reglas de negocio (*DRL - Definition Rules Language*. Tiene como premisa principal decir a las aplicaciones el "Que" hacer mas no el "Cómo".

Para su funcionamiento, Drools extiende y hace uso del algoritmo *RETE*, el cual permite la ejecución eficiente de condicionantes tipo *SI ... ENTONCES* haciendo uso de un modelo Orientado a Objetos. Además, facilita un estilo declarativo de programación, haciendo coincidir las estructuras de datos manejadas con la semántica del dominio del problema. Además, a partir de la versión 6, Drools introduce un nuevo algoritmo denominado *PHREAK*, el cual introduce una serie de optimizaciones y rediseños al algoritmo RETE, mejorando de manera considerable la eficiencia y rendimiento del algoritmo de transformación de reglas Salatino, Aliverti y De Maio, 2016, pg. 13.

Desde la versión 6, Drools se integra con otros productos de la familia JBoss enfocados en la automatización de aspectos del negocio, tales como jBPM y Planner. La iniciativa se denomina *KIE* (Knowledge is Everything), la cual brinda una base unificada para la administración del conocimiento del negocio Jboss, 2016a. La integración de productos permite, entre otras cosas, que las reglas de negocio puedan ser integradas con otras herramientas tales como: automatización de flujos de trabajo (jBPM), planeador de recursos (Opta-Planner), construcción de eventos complejos (Drools Fusion) o la construcción de Dashboards personalizados (Dashbuilder)[5].

Entre sus ventajas se encuentran:

- Ofrece varios mecanismos de construcción de reglas, entre las que se encuentran el uso del DSL denominado DRL, uso de hojas de Excel o construcción gráfica de las reglas usando Drools Workbench
- Fácil integración con otras herramientas para manejo del conocimiento empresarial (por medio de la iniciativa KIE)
- Drools posee una comunidad activa, además de soporte comercial por medio de JBoss/Red Hat Corporation.

[5]La descripción de estas aplicaciones se puede encontrar en http://www.kiegroup.org/

Evaluacion de Soluciones BRMS

De acuerdo a las opciones de BRMS descritas previamente, se realizó una evaluación para determinar cual Motor de Reglas sería el mas conveniente para implementación del presente proyecto. En la Tabla 4.1 se describe los criterios para evaluación de los BRMS, al igual que el peso que tiene cada criterio de acuerdo a la importancia para el desarrollo del trabajo de grado.

Tabla 4.1: Criterios de selección del BRMS

Criterios de Selección	Peso
API de integración con entornos de desarrollo	40 %
Facilidad de uso del Lenguaje de Reglas	30 %
Disponibilidad de servidores de Reglas de Negocio	15 %
Facilidad de uso por parte de personas del negocio	15 %

La evaluación de cada uno de los BRMS expuestos se muestran en las tablas 4.3 a 4.5.

Tabla 4.3: Evaluación BRMS *OpenRules*

Criterios de Selección	Peso	Puntaje	Ponderado
API de integración con entornos de desarrollo	40 %	5	2.0
OpenRules posee un API para Java amigable, compuesta por dos clases (*Decision* y *Response*) que permiten la captura de la salida del motor de reglas.			
Facilidad de uso del Lenguaje de Reglas	30 %	3	0.9
El motor no suministra un lenguaje propio de reglas de negocio, sino que ofrece una serie de convenciones y formatos para configurar archivos en Excel con el fin de que puedan ser interpretados correctamente. Existe una alta dependencia con el archivo(s) de reglas y su correcta estructura.			
Disponibilidad de servidores de Reglas de Negocio	15 %	3	0.45
Permite establecer repositorios de reglas de negocio a través de un repositorio de archivos en Excel que puede ser accedido mediante URL, dando la libertad al negocio de utilizar sistemas de almacenamiento externos. Sin embargo, para la ejecución de las reglas, no dispone de un servidor que cumpla con tal función, siendo necesario la implementación manual de un Web Service que cumpla con dicho fin (En la página de OpenRules suministran un manual de cómo configurar el motor de ejecución en modo de Web Sevice.).			
Facilidad de uso por parte de personas del negocio	15 %	5	0.75
Esta es una de las características mas fuertes del BRMS, dado que toda la administración de las reglas se centra en archivos de Excel, los cuales son de fácil acceso y manipulación por parte del personal del negocio.			
TOTAL			4.10

Tabla 4.4: Evaluación BRMS *Esper*

Criterios de Selección	Peso	Puntaje	Ponderado
API de integración con entornos de desarrollo	40 %	5	2.0
Esper está pensado desde un inicio para el desarrollo en Java, brindando una API amigable para la detección de eventos complejos y posterior lanzamiento de Listeners (Objetos POJO).			
Facilidad de uso del Lenguaje de Reglas	30 %	3	0.9
El Motor ofrece un lenguaje denominado EPL - Event Processing Language, el cual facilita la detección de eventos a capturar haciendo uso de un lenguaje similar a SQL. Sin embargo, la definición de las reglas debe realizarse a nivel de código, limitando la administración y mantenimiento únicamente por personas con conocimiento en desarrollo.			
Disponibilidad de servidores de Reglas de Negocio	15 %	2	0.3
No posee de un servidor de reglas de negocio en su versión Open Source. En la versión comercial ofrece integración con servidores de Aplicación para implementar Web Services para consumo de reglas negocio.			
Facilidad de uso por parte de personas del negocio	15 %	1	0.15
En su versión Open Source, no ofrece un mecanismo que permita al usuario final modificar las reglas establecidas. La modificación de las reglas se debe realizar modificacando código.			
TOTAL			3.35

Tabla 4.5: Evaluación BRMS *Drools*

Criterios de Selección	Peso	Puntaje	Ponderado
API de integración con entornos de desarrollo	40 %	5	2.0
Drools cuenta con un API amigable para el trabajo en Java, ofreciendo varias alternativas para la ejecución de reglas a través de la interfaz *KIE*.			
Facilidad de uso del Lenguaje de Reglas	30 %	5	1.5
El Motor ofrece un lenguaje denominado DRL, cuya sintaxis permite la captura de cualquier condición que ocurra en un grupo de Hechos (Facts, u objetos Java), además de ejecutar acciones predefinidas en el modelo de entidades definida por el negocio.			
Disponibilidad de servidores de Reglas de Negocio	15 %	5	0.75
Posee un repositorio de reglas denominado *Drools Workbench*, desde el cual se puede realizar administración de las reglas de negocio, y centralizando su acceso a través de un repositorio tipo *Maven*. Aunque no cuenta con un servidor para ejecución de reglas de negocio, a través de otra aplicación se puede generar las reglas de negocio en formato DRL y ser transformadas en Web Services (usando *WSO2 Business Rules Server*).			
Facilidad de uso por parte de personas del negocio	15 %	4	0.6
Aunque el uso del lenguaje DRL puede ser dificil al inicio para el manejo de reglas, posee una sintaxis sencilla y amigable lo cual facilita su aprendizaje. Por otra parte, la generación de reglas a partir de tablas de decisión basadas en Excel facilita a las personas de negocio el mantenimiento de reglas.			
TOTAL			4.85

De acuerdo a las evaluaciones realizadas, Drools es escogido como Motor de Reglas de Negocio para el desarrollo del presente proyecto. Las razones que llevan a escoger dicho Motor, además de las mencionadas, es que permite acceso a atributos y métodos de los objetos disponibles en un conjunto de clases. Dicha característica es importante al momento de realizar la integración con el framework de automatización de pruebas, dado que los objetos que representan cada una de las vistas del Software Bajo Pruebas (*SUT* - Software Under Test) puede ser accedida como Hecho (Fact) desde los archivos de definición de reglas de negocio.

Descripción de la implementacíon del BRMS en el proyecto

Para la implementación del BRMS, se utilizó como lenguaje de reglas de negocio el estándar manejado en el mercado (Drools Rules Language). El uso de dicho estándar permite crear reglas en el formato WHEN (Cuando) ... THEN (Espero), además de definir acciones a realizar sobre un modelo de datos o *Facts*.

Para la ejeución de las reglas definidas, fue necesario la instalación un motor de reglas de negocio. Para el presente prototipo, se exploraron dos herramientas que utilizan Drools como motor de ejecución de reglas de negocio:

- *Drools Workbench*, la cual es la herramienta soportada por los creadores de Drools, y que permite la administración centralizada de las reglas de negocio. Aunque ofrece buenas herramientas para temas de control de versiones y organización del repositorio de reglas, no ofrece una alternativa centralizada como motor de ejecución de reglas, siendo necesario la instalación de una librería en el cliente para ejecutar las reglas.

- *WSO2 Business Rules Server*, el cual es un servidor SOA que ayuda a exponer las reglas de negocio como si se tratase de un Web Service, de tal manera que al ser consumido ejecuta las reglas de negocio en el motor y expone una respuesta. Debido a que suministra herramientas gráficas tipo Wizard para la administración de reglas y su facilidad de integración (gracias a su API SOAP), esta herramienta fue escogida como solución BRMS para el presente proyecto.

4.1.3. Selección del Framework de Automatización de Pruebas

Actualmente, existen varios productos de tipo Comercial y Open Source que ofrecen un marco de trabajo para la ejecución de pruebas. Algunos de ellos ofrecen mas características funcionales, permitiendo que una persona con pocos concimientos técnicos pueda implementar fácilmente los casos de prueba automatizados. Sin embargo, existen acciones que no pueden ser realizadas únicamente con la experiencia funcional del SUT, por tanto es importante contar con un framework que ofrezca una API robusta de integración con Lenguajes de Programación.

Para el presente trabajo de grado y con el fin de evitar problemas de Licenciamiento, se decidió utilizar únicamente herramientas Open Source para la implementación de una solución de Framework Automatización. Frente a dicho alcance, se encontraron un total de 3 herramientas, las cuales fueron evaluadas de acuerdo a los criterios de selección que se muestran en la Tabla 4.6.

A continuación se realizará una descripción de cada herramienta y sus principales ventajas / desventajas. Luego de ello, se procederá a dar una ponderación de cada herramienta de acuerdo a los criterios establecidos.

Tabla 4.6: Criterios de selección del Framework de Automatización de Pruebas

Criterios de Selección	Peso
API de programación de Scripts	35 %
Identificación de Objetos	25 %
Escabilidad de los Scripts y facilidad de mantenimiento	15 %
Soporte en varios exploradores	5 %
Desarrollo activo	5 %
Recorder	5 %
Reportes	5 %
Soporte	5 %

Watir

Watir[6] es una libreria hecha en Ruby que permite la interacción con varios tipos de Browsers, y realiza una interacción con cada uno de los elementos de un sitio web como si de un usuario real se tratase. En sus inicios, el framework fue diseñado para soportar únicamente Internet Explorer, pero en las últimas versiones ha venido ampliando su soporte a otros browsers, teniendo al día de hoy compatibilidad con Firefox, Chrome y Safari.

Entre sus ventajas se encuentra:

- API amigable y simple de usar
- Ejecución de pruebas sin requerir de una interfaz gráfica
- Fácil integración con motores de ejecución de pruebas (específicamente con *Selenium Webdriver*.

Entre sus desventajas se encuentran:

- Crear versiones de Watir que soporten otro tipo de browsers no es sencilla ni simple.
- Mantener el API de Watir con las nuevas versiones de los exploradores no es sencilla, teniendo un tiempo considerable entre el lanzamiento de una versión de browser y una nueva versión de Watir.

[6] http://watir.github.io/

Sahi

Sahi[7] es una herramienta con soporte Open Source y Comercial, que surgió en el año 2005 teniendo como principal enfoque la creación de una herramienta enfocada a los testers, teniendo en cuenta que otras soluciones tenían como principal consumidor los desarrolladores. Frente a ello, en Sahi nos encontramos con un potente recorder capaz de grabar cualquier tipo de acción de un sitio Web, desde acciones sobre HTML estándar hasta en Flash (Flex), applets, etc.

Entre sus ventajas se encuentra:

- Ofrece un grabador de scripts (conocido como *Recorder*) que funciona en cualquier browser, generando además scripts compatibles con cualquier navegador
- Facilita la ubicación inteligente de identificadores dentro de un sitio Web, aspecto importante al momento de presentarse cualquier tipo de cambio en la interfaz gráfica del sitio, ya que suministra ubicaciones relativas (ej: _near, _in, etc) de tal manera que no se ve afectado al momento de cambio de algún elemento o su identificador.
- Suministra un soporte nativo a los tiempos de espera de las peticiones AJAX, de tal manera que no hay necesidad de emplear código adicional para dar dicho soporte

Entre sus desventajas se encuentran:

- Para el desarrollo de los scripts, es necesario el uso de su propio lenguaje de scripts (denominado *Sahi Script*). Aunque posee una sintaxis similar a la de Javascript, la ausencia de mecanismos de integración nativa con otros lengaujes de programacion dificulta la integración con otros mecanismos diferentes a los ofrecidos por el framework (por ejemplo, un motor de reglas de negocio).
- Aunque el uso de ubicaciones relativas puede facilitar la labor de mantenimiento, a la vez puede volver ambigua la prueba aplicada. Lo anterior dado que Sahi retorna solo el primer elemento que encuentre, y es muy probable que elementos relativos dentro de la página Web puedan aparecer repetidos.
- Aunque tiene soporte comercial, la comunidad no es tan grande como la que tienen otros Frameworks.
- Ofrece algunas características solo en su licencia comercial (ej: tomar pantallazos o ejecución en paralelo) que otras herramientas ofrecen en su licenciamiento Open Source.

[7] http://sahipro.com/

SeleniumHQ

SeleniumHQ[8] es un proyecto de automatización de interacción en browsers, diseñado inicialmente como herramienta de automatización de pruebas en 2004. Desde entonces, el proyecto ha ido lanzando varios productos que facilitan el desarrollo de scripts de automatización, con compatibilidad con varios lenguajes de programación (Java, C#, PHP, Javascript, etc). Además, cuenta con varios sponsors que dan soporte económico y técnico al proyecto, y otras organizaciones conocidas que utilizan el producto (ej: Google, Linkedin, ThoughWorks).

Entre sus ventajas se encuentran:

- Cuenta con varios proyectos que permiten que distintos roles de un proyecto (testers y desarrolladores) puedan crear scripts de pruebas
- Permite la ejecución de pruebas sin necesidad de una interfaz gráfica del lado del browser (*headless*)
- Tiene excelente compatibilidad con certificados HTTPS ya que no hay necesidad de instalar un certificado propietario sino utiliza el nativo de cada sitio Web.
- Tiene una comunidad activa, además de contar con varios Sponsors[9] que aportan a la base de código.

Entre sus desventajas se encuentran:

- No tiene buen soporte para todas las versiones de Internet Explorer
- Al momento en que cambia la posicion en pantalla de algún elemento en pantalla, se puede perder la referencia del mismo en el script, generando inconvenientes al momento de ejecución
- Es necesario establecer controles de espera frente a la aparición de los elementos con los cuales se va a tener interacción. Esto presenta dificultades en especial en peticiones tipo AJAX.
- Su *recorder* solo es soportado en Firefox.

[8] http://www.seleniumhq.org/
[9] http://www.seleniumhq.org/sponsors/

Análisis comparativo de las herramientas descritas

De acuerdo a las características expuestas de cada framework, se procedió a brindar una ponderación de 1 a 5 de cada solución, siendo 1 la mas baja y 5 la mas alta. Dicha ponderación se multiplicó por cada criterio establecido en la tabla 4.6 con el fin de establecer cual solución sería la mas apropiada para el presente trabajo de grado. El análisis y resultados aplicados de acuerdo a los criterios son mostrados en las Tablas 4.8 a 4.10.

Tabla 4.8: Evaluación del Framework de Automatización *Watir*

Criterios de Selección	Peso	Puntaje	Ponderado
API de programación de Scripts	35 %	4	1.4
Watir soporta únicamente a Ruby como Lenguaje de programación, aunque por otra parte suministra una API amigable para el desarrollador.			
Identificación de Objetos	25 %	3	0.75
Permite la identificación de objetos (o elementos) mediante id o name			
Escabilidad de los Scripts y facilidad de mantenimiento	15 %	3	0.45
Debido a que tiene un API amigable, facilita la legibilidad del código para futuros mantenimientos. Sin embargo si el sitio Web llega a presentar cambios extremos en su estructura o identificadores, el script requerirá modificaciones para ajustarse.			
Soporte en varios exploradores	5 %	4	0.2
Debido a que Watir es una capa de API que accede a motores de ejecución de scripts de pruebas, puede usar otros frameworks como motor (por ejemplo Selenium). Por tanto, a través de dichos motores tiene compatibilidad con todos los browsers disponibles.			
Desarrollo activo	5 %	3	0.15
Al revisar su repositorio de código (https://github.com/watir/watir) se identifica que hay aportes constantes al código fuente. Sin embargo, dichas contribuciones vienen de dos personas, mostrando que probablemente no es un desarrrollo tan activo para la comunidad.			
Recorder	5 %	1	0.05
No soporta un Recorder nativo que permita una grabación desde el browser y exporte a instrucciones en Watir.			
Reportes	5 %	1	0.05
No posee una forma nativa de realizar soportes			
Soporte	5 %	2	0.1
Ofrece un canal en Stackoverflow y un grupo de email para preguntas de soporte. No existe una empresa que brinde soporte comercial.			
TOTAL			3.15

Tabla 4.9: Evaluación del Framework de Automatización *Sahi*

Criterios de Selección	Peso	Puntaje	Ponderado
API de programación de Scripts	35 %	5	1.75
Sahi soporta Javascript como lenguaje para construir los scripts. El API utilizada para interacción con el browser es amigable, además que permite ser integrada con otros lenguajes (ej: Java) Lamentablemente, la mayoría de funciones que en otros frameworks son sin ningún costo adicional, en Sahi solo son soportadas en su versión comercial (ej: ejecución distribuida de tests, editor de scripts, tomar pantallazos, aceptar certificados SSL)			
Identificación de Objetos	25 %	4	1.0
El API utilizada permite identificar elementos mediante name o id. Además, el uso del Recorder facilita la identificación de cada uno de los elementos.			
Escabilidad de los Scripts y facilidad de mantenimiento	15 %	3	0.45
En algunos casos el script se puede acomodar a los cambios presentados en una aplicación, dado que permite el uso de selectores de cercania con elementos existentes. Lamentablemente, esta es una carácterística soportada por la versión Comercial.			
Soporte en varios exploradores	5 %	5	0.25
Tiene soporte en todos los browser populares			
Desarrollo activo	5 %	3	0.15
Al revisar el código fuente de la versión Open Source (https://sourceforge.net/p/sahi/code) no se identifica que la última fecha de modificacion del repositorio fue de 2014. Sin embargo, a partir de la versión 5.1 el código fuente se anexa como una descarga adicional al binario de la aplicación, la cual si tiene una fecha reciente (2016).			
Recorder	5 %	3	0.15
El recorder es uno de los mas completos de las tres herramientas evaluadas, ya que permite realizar grabaciones desde cualquier explorador. Lamentablemente, la versión Open Source no tiene todas las características que soporta el Recorder, por lo que será necesario realizar modificaciones manuales a los scripts.			
Reportes	5 %	5	0.25
Posee un mecanismo nativo para generar reportes de validación de pruebas que fueron escritas en el script.			
Soporte	5 %	3	0.15
Ofrecen soporte a través de la lista de correo, pero no es mucho para usuarios gratuitos. Por otra parte suministran documentación detallada en su sitio de ayuda.			
TOTAL			4.15

Tabla 4.10: Evaluación del Framework de Automatización *SeleniumHQ*

Criterios de Selección	Peso	Puntaje	Ponderado
API de programación de Scripts	35 %	5	1.75
Ofrece una API que es compatible con varios lenguajes de programación (Java, PHP, C#, PHP).			
Identificación de Objetos	25 %	5	1.25
El API permite identificación por varios mecanismos disponibles en el browser (id, cualquier atributo, selector css, nombre de clase, xpath, nombre de link, contenido parcial de un link).			
Escabilidad de los Scripts y facilidad de mantenimiento	15 %	3	0.45
El API es compatible para el manejo del patrón de diseño PageObject, lo cual facilita la mantenibilidad del código. Sin embargo, se identifica una deficiencia en el volumen de código a ser mantenido cuando se trata de aplicaciones con varias vistas, ya que incrementa el número de Page Objects a ser codificados.			
Soporte en varios exploradores	5 %	5	0.25
Tiene soporte en todos los browser populares			
Desarrollo activo	5 %	5	0.25
Al revisar el código fuente (https://github.com/SeleniumHQ/selenium/commits/master) tiene constantes actualizaciones sobre su base de código en rango de 2 - 3 horas. Además, hay varios desarrolladores realizando aportes al repositorio, lo cual demuestra que se encuentra activo en la comunidad.			
Recorder	5 %	4	0.2
Aunque el recorder solo es soportado en Firefox, permite la identificación de los selectores mínimos necesarios para comenzar la automatización de pruebas. Además, ofrece la posibilidad de extender su funcionalidad básica mediante plugins.			
Reportes	5 %	4	0.2
No posee un mecanismo nativo de Soporte. Sin embargo, presenta una buena integración con herramientas de pruebas unitarias.			
Soporte	5 %	5	0.25
Existe un amplio soporte de la comunidad, por medio de: Grupo de Usuarios, Chat Room, Bug Tracker, además de ofrecer soporte comercial a través de sus sponsors.			
TOTAL			4.6

De acuerdo al análisis realizado, se llega a la conclusión que para el presente proyecto de grado se escogerá SeleniumHQ como framework de automatización de pruebas, debido a sus características fuertes como son su API amigable, los mecanismos que tiene para identificación de objetos, su soporte en varios exploradores y su facilidad de integración con frameworks de pruebas unitarias y con otras librerias (como por ejemplo, un Motor de Reglas de Negocio).

Implementación del Framework seleccionado en el proyecto

Selenium se encuentra compuesta por tres componentes:

- IDE, el cual es un complemento para Firefox que permite la grabación de las acciones que realiza un usuario sobre una página Web.
- WebDriver, el cual es un API utilizado en un lenguaje de desarrollo (Java, Python, C#, etc) con el fin de agregar elementos de progamación a los casos de prueba base, que no pueden ser soportados por el IDE.
- Grid/Hub, el cual es un servidor que permite la administración de varios nodos de browser a los cuales se pueden enviar las pruebas de una forma centralizada

Respecto a los componentes mencionados, para el presente proyecto se utilizó *WebDriver* y *Grid/Hub*. Para el caso del uso de WebDriver se encontraron diferentes formas de uso que en muchos casos conducían a la producción de código de baja calidad o de difícil mantenimiento, por lo cual luego de realizar investigaciones sobre mejores prácticas se determinó que el uso del patrón de diseño de pruebas *Page Objects* en el diseño de las pruebas.

Por otra parte, para la administración de las distintas instancias de browsers que ejecutarán las pruebas, se utilizó *Selenium Grid/Hub*. Hub actua como un receptor/centralizador de todas las peticiones de ejecución de pruebas enviadas por los Page Objects, y las distribuye a uno o mas nodos que pueden ejecutar diferentes versiones de los exploradores de Internet (Chrome, Firefox, Internet Explorer, entre otros).

La infraestructura empleada por el presente prototipo para el montaje de Selenium se puede observar en la capa de infraestructura de la Figura 4.1.

4.1.4. Generación de Reportes

Con el fin de que los scripts de pruebas pudiesen ser ejecutados con regularidad después de todo cambio realizado en el componente de *Confirmación Clases*, se implementó la ejecución de las mismas como pruebas unitarias utilizando JUnit. El uso de pruebas unitarias facilita la generación de reportes de cobertura de pruebas, ayudando a los desarrolladores a identificar las partes del código que presentaron fallas después de un cambio o nueva funcionalidad agregada al Software.

Los reportes sobre la cobertura en pruebas ofrecidos por las pruebas unitarias pueden ser vistos de dos formas: a través del IDE de desarrollo, o mediante el uso de servidores de integración contínua. Para el caso del presente proyecto, se utilizó Jenkins como servidor de integración contínua, debido a que facilita un acceso centralizado a los resultados de las ejecuciones de las pruebas realizadas, además de guardar un histórico de ejecuciones de las mismas con sus respectivos resultados.

4.2. Exploración de otras alternativas de solución

Tal como se exploró en la sección 2.6 sobre BPM, dicha metodología junto con el lenguaje BPMN pueden suministrar una solución viable para el problema planteado. Se llego a considerar su integración dentro de la arquitectura propuesta debido a la facilidad de creación de diagramas de flujo de tareas utilizando software disponible para dichos fines. Sin embargo, dado que el propósito de BPM está enfocado en la optimización de procesos de negocio y creación de flujos de actividades que pueden llegar a ser semi-automáticos, se descartó su integración, dado que el propósito del presente proyecto está mas enlazado con demostrar la integración de las reglas de negocio con un flujo ya establecido (el suministrado por el SUT) mas no construir un nuevo flujo utilizando una notación (BPMN).

4.3. Alcances y Limitaciones de la solucion propuesta

De acuerdo a las bases expuestas en los capítulos previos sobre reglas de negocio y automatización de pruebas, se identifica una posible integración de ambos conceptos. Las reglas de negocio podrían servir como repositorio dinámico de datos de prueba de los cuales los scripts de prueba pueden alimentarse para marcar una vista de una aplicación Web específica como válida/inválida.

El tipo de reglas de negocio empleados para esta integración son de Tipo Operativo. Sin embargo difieren con la definición estricta de regla Operativa, toda vez que no se pretente forzar o restringir estados en un sistema sino verificar la existencia de una condición, conllevando a la acpetación o rechazo de los respectivos hechos (facts) introducidos en el motor de reglas.

Por otra parte, los motores de reglas de negocio solo pueden verificar la existencia de condiciones para proceder con la ejecucion de acciones ya definidas o modificación/creación de datos sobre metadatos existentes. La anterior condicición implica que en ningún momento una regla de negocio podrá agregar acciones o interacciones nuevas al script de pruebas.

Por tanto, los scripts de pruebas definidos para cada aplicación solo tendrán un conjunto de acciones y verificaciones delimitadas por el estado de las vistas en un momento dado. Si las vistas llegasen a cambiar y se llegara a agregar nuevos campos o información suceptible de validar, es necesario ajustar los scripts de pruebas.

Sin embargo, cuando se presenten cambios en información o datos propios que arroja la aplicación (ejemplo: nombres o apellidos de una persona, resultados de ejecución de una transacción bancaria o total a pagar por una factura), las reglas de negocio podrán suministrar un mecanismo ajustable para modificar la validación de dichos datos, sin implicar la modificación directa de los scripts de prueba.

CAPÍTULO 5
Ejecución

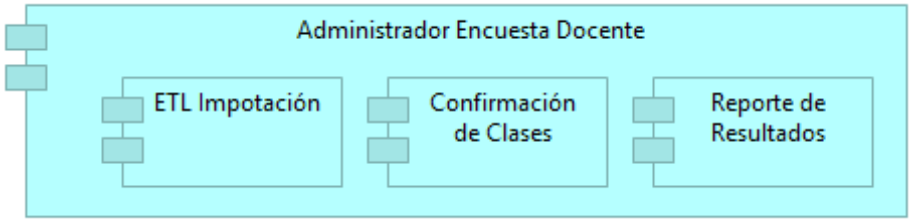
5.1. Análisis

5.1.1. Descripción general del Software Bajo Prueba (SUT)

Para realizar el presente proyecto, se partió de un análisis del Software Bajo Prueba (SUT) cuyas pruebas se pretendía automatizar. El SUT que se escogió fue el Software de Encuesta Docente, el cual es utilizado por una Institución de Educación Superior para realizar la programación de encuestas sobre las clases tomadas por los estudiantes en un semestre específico.

Dicho software se encuentra compuesto en tres componentes principales, los cuales se pueden visualizar en la Figura 5.1:

Figura 5.1: Arquitectura de la Aplicación *Administrador Encuesta Docente*



Fuente: Autor

- *ETL de Importación*, el cual es utilizado para importar la información acerca de docentes, estudiantes y clases desde el Sistema Académico de la Universidad,
- *Confirmación de Clases*, el cual es un módulo al cual tiene acceso los Directores y delegados de los departamentos académicos de la Universidad con el fin de confirmar si una clase/docente debe ser evaluada por un estudiante.
- *Reporte de resultados*, el cual permite visualizar de una forma consolidada las respuestas dadas por los estudiantes frente a las encuestas programadas.

Para el desarrollo del prototipo, se escoge la automatización de pruebas del componente de *Confirmación de clases*.

5.1.2. Análisis de las vistas del SUT

El flujo en el cual funciona el componente de *Confirmación Clases* es descrito en la Tabla 5.1.

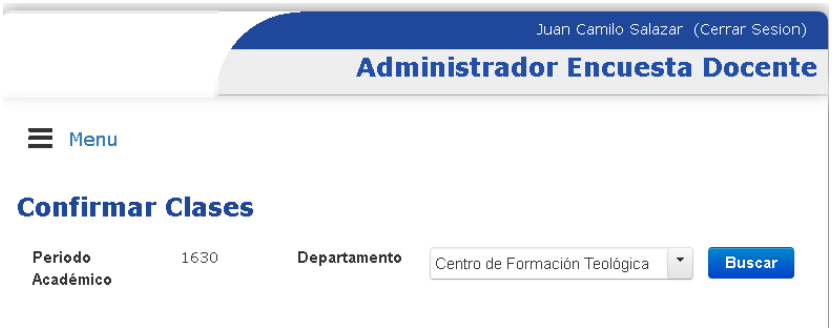
Tabla 5.1: Descripción del Caso de Uso del componente *Confirmación de Clases*

Usuario	**Sistema**
Ingresa al Link *Confirmar Clases*	Muestra una vista con el número de período académico actual y una lista de departamentos, de acuerdo a los departamentos a los cuales tiene permiso acceder el usuario actual.
Selecciona un departamento de la lista y selecciona la acción de ”Buscar”	Muestra una vista con dos pestañas, cada pestaña con los siguientes títulos: *Sugeridas NO Evaluar*, *Sugeridas para Evaluar*. Cada pestaña muestra dos números, cada uno indicando el número de clases revisadas y sin revisar. Por defecto, muestra una lista de clases en la pestaña de *Sugeridas NO Evaluar*, cada item de la lista con la siguiente información: Número Clase, Número Curso, Nombre Docente, Nombre Curso, No Estudiantes, y un botón que permite activar/desactivar la clase.
Selecciona una clase de la lista que se encuentra en la pestaña de *Sugeridas NO Evaluar* y activa la opción de *Confirmar*	Marca el estado de confirmación de la clase de ”No” a ”Si”
Selecciona la pestaña *Sugeridas para Evaluar*	Muestra una lista de clases, cada item de la lista con la información de la clase a confirmar, y al frente de cada clase una opción que permite cambiar el estado a ”No Confirmada”.
Selecciona una clase de la lista que se encuentra en la pestaña de *Sugeridas para Evaluar* y activa la opción de ”No Confirmar”	Marca el estado de confirmación de la clase de ”Si” a ”No”
Vuelve a seleccionar la acción de ”Buscar” en el menú de departamentos	Incrementa en 1 el indicador de *Clases Revisadas* de la pestaña de ”Sugeridas No Evaluar” Incrementa en 1 el indicador de *Clases Revisadas* de la pestaña de ”Sugeridas para Evaluar”

De acuerdo a lo señalado en la Tabla y de acuerdo a cada tipo de interacción esperada en el componente, se espera un total de 4 vistas diferentes que se pueden presentar en el software y por tanto cuyas pruebas son suceptibles de automatizar. Las vistas correspondientes son

mostradas en las Figuras 5.2 a la 5.5.

Figura 5.2: Vista Aplicación: Filtro de Búsqueda de Clases



Fuente: Autor

Figura 5.3: Vista Aplicación: Lista Clases Sugeridas NO Evaluar

Fuente: Autor

Figura 5.4: Vista Aplicación: Lista Clases Sugeridas a Evaluar

Fuente: Autor

Figura 5.5: Vista Aplicación: Indicadores de No Clases Revisadas

Fuente: Autor

5.1.3. Reglas de Negocio

De acuerdo al enfoque dado en la Sección 4.3, las reglas de negocio en las cuales se enfocó la fase de análisis fueron aquellas que verificaran el contenido de cada una de las vistas. En la tabla 5.2 se exponen el conjunto de reglas que se validarán por cada vista.

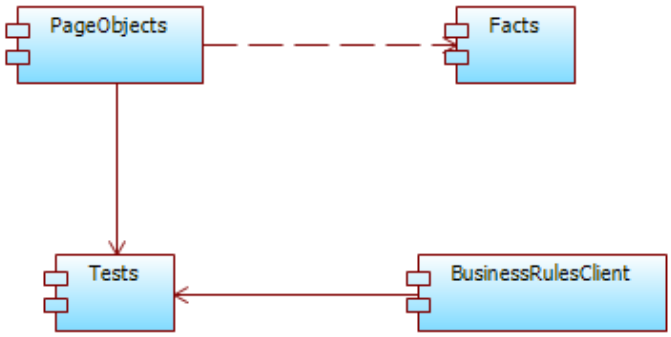
Tabla 5.2: Reglas de Negocio para Componente *Confirmar Clases*

Vista	Regla de Negocio
Búsqueda de Clases	**Cuando:** el número de departamentos en la lista de departamentos sea diferente a 126 **Espero:** que se arroje un error con el texto ″El número de departamentos esperados es 126″
	Cuando: el periodo académico que aparece en pantalla sea diferente a 1630 **Espero:** que se arroje un error con el texto ″El período académico al cual aplica las pruebas actuales es 1630″
	Cuando: la vista de Busqueda de clases se encuentra abierta **Espero:** que se seleccione en la lista desplegable de departamentos la opción ″Centro de Formación Teológica″
Clases Sugeridas NO Evaluar	**Cuando:** el número de clases que aparece en la tabla sea diferente a 0 **Espero:** que se arroje un error con el texto ″Se espera un total de clases que se han Sugerido para NO Evaluar igual a 0″
Clases Sugeridas para Evaluar	**Cuando:** el número de clases que aparece en la tabla sea menor a 159 **Espero:** que se arroje un error con el texto ″Se espera un total de clases confirmadas mayor o igual a 159″
	Cuando: el código de curso es 3275 Y el código de clase es 4032 Y el docente sea Arango Alzate Oscar **Espero:** que la clase se marque como confirmada
	Cuando: el código de curso es 3275 Y el código de clase es 4038 Y el docente sea Martinez Morales Dario Ernesto **Espero:** que la clase se marque como confirmada
Actualización Busqueda de Clases	**Cuando:** en la pestaña de Sugeridas NO Evaluar el indicador de Revisadas"sea diferente a 0 **Espero:** que se arroje un error con el texto "Se espera un no de clases Revisadas igual a 0 en la pestaña Sugeridas NO Evaluar"
	Cuando: en la pestaña de Sugeridas para Evaluar el indicador de Revisadas"sea diferente a 1 **Espero:** que se arroje un error con el texto "Se espera un no de clases Revisadas igual a 1 en la pestaña de Sugeridas para Evaluar"

5.2. Implementación

Para el diseño de los scripts de pruebas, se decidió utilizar la estructura de componentes que se muestra en la Figura 5.6. La función que desempeña cada componente es descrita en la Tabla 5.4.

Figura 5.6: Estructura de Componentes de la Aplicación



Fuente: Autor

Tabla 5.4: Función de los componentes de la aplicación de automatización de pruebas

Componente	Función
PageObjects	Agrupa los denominados scripts de prueba, que son los que van a tener interacción directa con la aplicación Web (en nuestro caso con el app de Encuesta Docente)
Facts	Conjunto de clases POJO's (Plain Old Java Objects) que son utilizados para la comunicación entre PageObjects y BusinessRules (reglas de negocio)
BusinessRules Client	Conjunto de clases que se comunican con el Sistema de Administración de Reglas de Negocio (BRMS) *WSO2 Business Rules Server*. Estas clases actuan como clientes del Web Service expuesto por el BRMS.
Tests	Clases de Pruebas Unitarias (JUnit) que se encargan de la implementación de la parte dinámica de las pruebas, es decir, interactúan con los PageObjects para veejecutar acciones sobre el *SUT*, luego obtienen la interpretación de lo que se muestra en la vista usando las clases POJO's en el componente *Facts*, los cuales pasa a validar a través del motor de reglas de negocio (usando el componente *BusinessRulesClient*).

5.2.1. Configuración de Scripts de Prueba

Respecto a la codificación, se empleó Java como lenguaje de programación, además de la librería *Selenium WebDriver* como framework de Automatización de Pruebas. Se creó un total de 5 clases correspondientes a cada una de las páginas o vistas con las cuales se realiza interacción, utilizando el patrón de diseño *Page Object*.

Para la ejecución de los scripts creados, se procedió a configurar la herramienta *Selenium Hub*, usando un nodo del Browser Mozilla Firefox comunicado con el Hub. La instalación y configuración se realiza utilizando Docker [1], y descargando las imágenes oficiales ofrecidas por Selenium para la configuración del nodo Hub y de Firefox [2]. Como resultado final de la instalación, se puede observar la consola de administración de los nodos en el Hub (ver Figura 5.7)

Figura 5.7: Administración de Nodos de Selenium Hub

Fuente: Autor

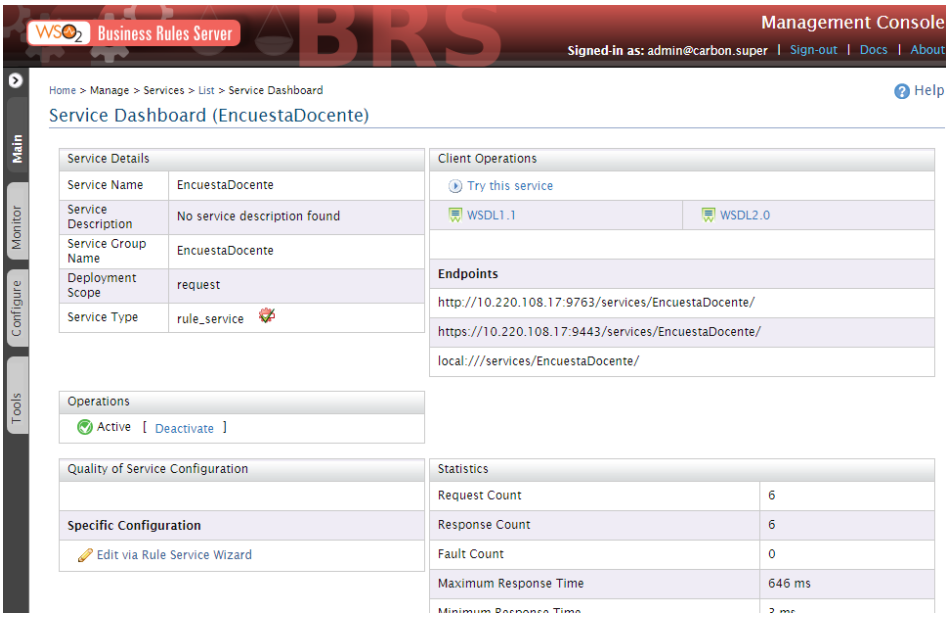
5.2.2. Configuración de Reglas de Negocio

Para la implementación de las reglas de negocio, se utilizó las clases establecidas en el componente de *Facts*, además de la creación de los archivos de reglas de negocio establecidas en la carpeta de recursos del proyecto usando el formato de reglas de negocio de Drools Jboss, 2016b. Tomando los artefactos mencionados, se construyó un archivo AAR de acuerdo a las instrucciones dadas por la documentación oficial de WSO2 WSO2 Inc., 2016, y se agrega al BRMS usando la interfaz administrativa del mismo. El archivo AAR contiene las instrucciones de las reglas de negocio, al igual que de los objetos de hechos (Facts) esperados tanto como entrada y salida del Web Service que se creará para las reglas parametrizadas.

De esta forma, se cargan las reglas de negocio a la interfaz administrativa de *WSO2 Business Rules Server*, en dónde ya quedan disponibles las reglas como un Servicio Web (ver Figura 5.8).

[1] https://www.docker.com/
[2] https://github.com/SeleniumHQ/docker-selenium

Figura 5.8: Administrador de configuración de Reglas Negocio - *WSO2 Business Rules Server*



Fuente: Autor

5.2.3. Configuración de Scripts de Prueba

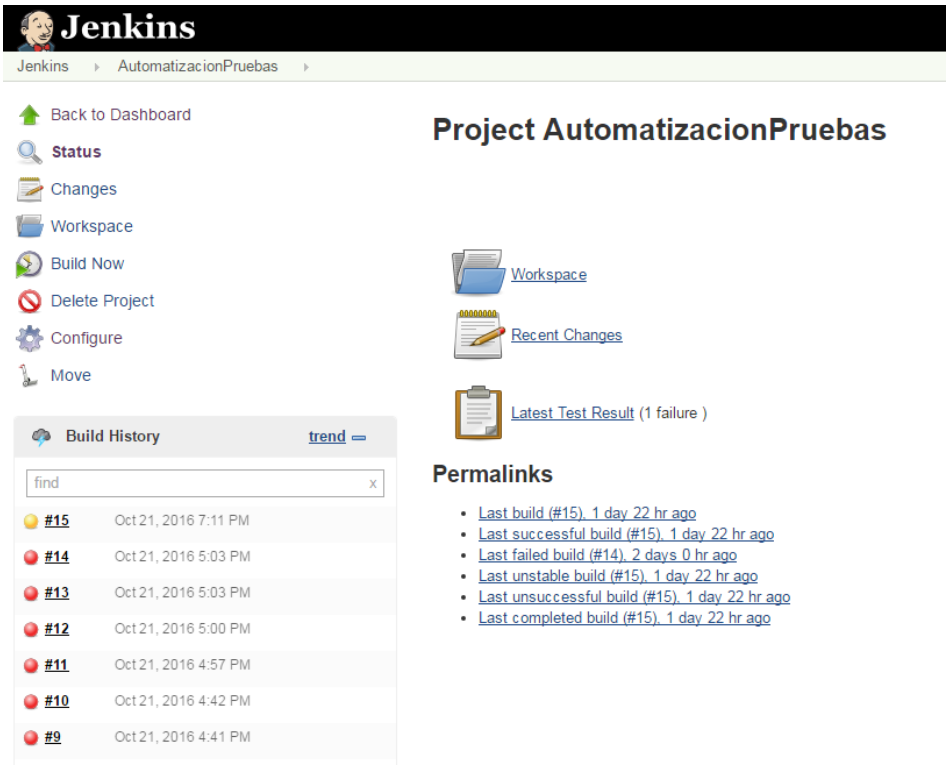
Para realizar la comunicación entre las reglas de negocio y los scripts de prueba, se implementó un cliente que se comunica con el Web Service que expone el BRMS. Para ello, se utiliza la herramienta *wsimport* suministrada con el JDK de Java, el cual genera las clases requeridas para el consumo.

Una vez realizada dicha configuración, se procede a crear las clases de pruebas unitarias, con las cuales se va creando cada uno de los escenarios de pruebas para verificar las reglas definidas en la tabla 5.2. La implementación de las clases de prueba se puede ver en la ruta *src/test/java* del proyecto.

5.2.4. Visualización de Resultados de las Pruebas

Con el fin de que las pruebas configuradas pudiesen ser ejecutadas con frecuencia y sus resultados almacenados históricamente y posteriormente consultados, se procedió a configurar el servidor de integración contínua *jenkins*. Para ello, se utilizó una imagen de Docker para la instalación de la herramienta, y luego se procedió a configurar el código fuente, la rutina de compilación y posterior generación de reportes por las opciones administrativas de la herramienta. Finalmente, se puede ejecutar las pruebas de forma contínua a través de la interfaz administrativa de la herramienta, tal como se puede visualizar en la Figura 5.9.

Figura 5.9: Administración de Ejecuciones del Servidor Integración Contínua Jenkins



Fuente: Autor

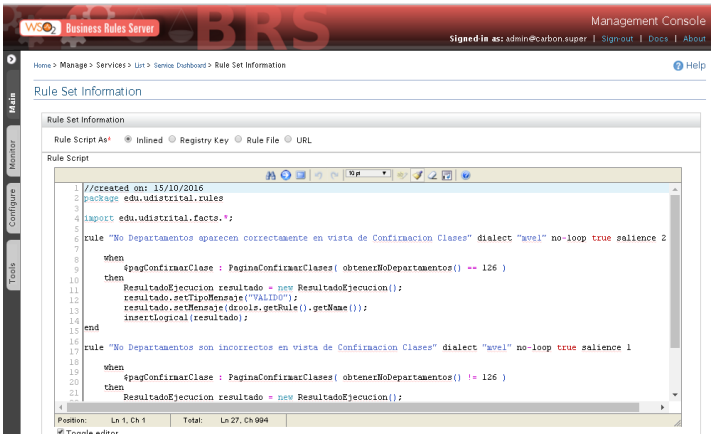
PARTE III

CIERRE DE LA INVESTIGACIÓN

CAPÍTULO 6
Resultados y Discusión

A partir del diseño definido previamente (Capítulo 4) y su aplicación sobre el caso de estudio explicado en el Capítulo 5, se logró crear un prototipo funcional que cumplía con los requisitos exigidos por los casos de prueba y a su vez demostró que con la implementación de la integración entre un BRMS y un Framework de Automatización de Pruebas es posible mantener la separación de los datos y validaciones de una prueba de su lógica de ejecución. La elaboración de las reglas de negocio enfocadas a la validación de datos, aunque requirieron de cierto conocimiento en el lenguaje DSL [1] propuesto por el BRMS Drools (y sobre el cual se basa WSO2 Business Rules Server), no implicaron mayor esfuerzo en su redacción, toda vez que se haya definido correctamente aquellos conceptos y validaciones clave que se debían realizar en las diferentes vistas soportadas por el SUT. Un ejemplo de las reglas definidas puede visualizarse en la Figura 6.1.

Figura 6.1: Edición de una regla de negocio sobre el BRMS



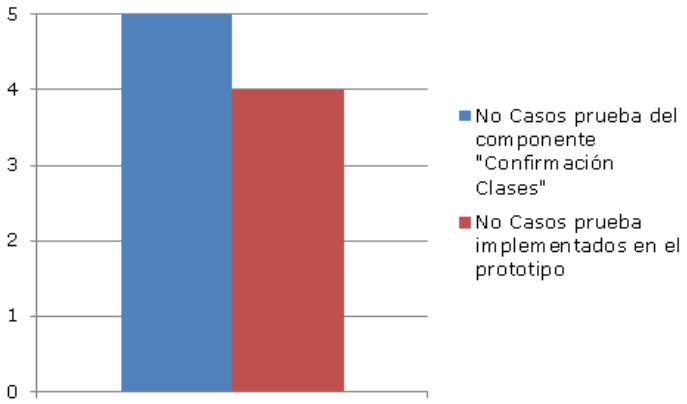
Fuente: Autor

Para la elaboración de las reglas de validación, fue clave la definición de los casos de prueba del componente del SUT evaluado. Dado que dichos casos de uso no estaban

[1] Domain Specific Language

definidos en la documentación del SUT, se procedió a realizar un levantamiento general delimitado al componente evaluado (*Confirmación de Clases*). La ejecución de esta actividad permitió identificar las diferentes vistas/pantallas y, por tanto, identificar los escenarios de ejecución de cada vista con sus respectivos datos esperados. El porcentaje de casos de prueba que fueron implementados se puede visualizar en la Figura 6.2. De acuerdo a dicha gráfica, quedó pendiente un caso de prueba (ver Anexo A.5) el cual no fue posible validar de forma automática con los scripts y por tanto requirió una validación manual.

Figura 6.2: Cubrimiento de los casos de prueba vs los scripts automatizados



Fuente: Autor

Por otra parte, se logró agregar al SUT evaluado la característica de Integración Contínua (IC), al hacer que las verificaciones implementadas por los scripts de prueba lograran ser ejecutadas por medio de tests unitarios y, por tanto, su ejecución y reproducción puedan ser repetibles después de cualquier cambio que se realice sobre el SUT. Las pruebas unitarias y la forma mediante la cual se puede visualizar su resultado de ejecución pueden ser visualizados en la Figura 6.3.

Figura 6.3: Ejemplo de visualización Resultados de la Ejecucion de los Tests

Fuente: Autor

CAPÍTULO 7
Conclusiones

7.1. Verificación, contraste y evaluación de los objetivos

Al contrastar los objetivos inicialmente propuestos, se identifica que se cumplió con la totalidad de los objetivos propuestos. El contraste entre los objetivos y su evidencia de cumplimiento se pueden visualizar en la Tabla .

Tabla 7.1: Contraste Objetivos vs Evidencias Cumplimiento

Objetivo	Evidencia de Cumplimiento
Identificar las herramientas y técnicas existentes en la actualidad para la implementación de motores de reglas de negocio, por medio de una investigación técnica de las opciones disponibles, para tenerlas en cuenta al momento de selección y despliegue de un motor de reglas de negocio.	En la descripción arquitectural del presente proyecto (Capítulo 4) se suministraron diferentes alternativas técnicas existentes que tenian como característica que fueran de caracter Open Source, finalmente escogiéndose para el proyecto el BRKMS *Drools*, utilizado bajo el servidor de Web Service *WSO2 Business Rules Server*.
Identificar las características de una implementación correcta de la automatización depruebas, por medio de la investigación de las buenas prácticas en este campo, paraasegurar la implementación exitosa del proceso de automatización.	La sección 3.3.3 describe las buenas prácticas que se deben tener al momento de implementar una solución de Automatización de Pruebas, de acuerdo al ISTQB, el cual es un instituto referente en la estandarización de los procesos de Pruebas en Software. Por otra parte, siguiendo un enfoque de diseño coherente con el descrito en la sección 3.3.2, se escogió *Page Objects*, que posee una gran aceptación por parte de la comunidad técnica. Por otra parte, se adaptó de los otros enfoques la buena práctica de separación de datos y lógica de validación, por medio de la implementación de dichos componentes en el BRMS y el Script de Pruebas, respectivamente.

Tabla 7.1: Contraste Objetivos vs Evidencias Cumplimiento (cont)

Objetivo	Evidencia de Cumplimiento
Implementar un prototipo de software para automatizar las pruebas del caso de estudio "Administrador de Encuesta Docente - Componente Confirmación Clases", por medio de una integración entre un motor de reglas de negocio y un framework de automatización de pruebas, para comprobar la validez de la integración.	Como se visualiza durante la ejecución del proyecto y los resultados, se pudo realizar la integración entre el motor de reglas de negocio y el framework de Automatización de Pruebas. El Software Under Test escogido para este proyecto ("Administrador de Encuesta Docente - Componente Confirmación Clases") logró demostrar que la mencionada integración es técnicamente posible y repetible para otros proyectos.

7.2. Síntesis del modelo propuesto

La síntesis del modelo propuesto se describe en el Capítulo 4, en dónde se describe con detalle la arquitectura aplicada en el desarrollo de la solución.

7.3. Aportes originales

Este proyecto propone la idea de que los scripts de automatización de pruebas pueden ser guiadas a través de las reglas de negocio establecidas en un Motor de Reglas de Negocio. De acuerdo a la exploración bibliográfica realizada, no existe un enfoque implementado similar al propuesto. Sin embargo, vale aclarar que el enfoque de Automatización de Pruebas presentado puede ser clasificado como un Framework de Automatización *Orientados a Datos*, tal cual como se describe en la sección 3.3.1.

CAPÍTULO 8
Prospectiva del Trabajo de Grado

8.1. Líneas de investigación futuras

El presente proyecto de grado presenta un punto de referencia, que de acuerdo a la investigación teórica realizada, no ha sido planteado previamente. Existe mucha investigación referente a reglas de negocio y casos de éxito de aplicación de la automatización de pruebas, inclusive con el enfoque *Orientado a Datos*. Sin embargo, la posibilidad de plantear una integración entre estos dos componentes no había sido explorada con anterioridad.

Por tanto, la integración de los frameworks de automatización de pruebas con otros componentes de negocio de gestión del conocimiento puede ser visto como una nueva linea de investigación. El objetivo de dicha línea sería explorar mejores formas de mejorar la mantenibilidad de los scripts de prueba, permitiendo su modificación dinámica a partir del conocimiento que tiene recolectado el negocio en las aplicaciones o software que poseen para la administración del conocimiento empresarial (como en este caso se exploró la integración con Motores de Reglas de Negocio).

8.2. Trabajos de Investigación Futuros

De acuerdo a la nueva línea de investigación propuesta, se pueden proponer los siguientes trabajos de investigación futuros:

- Integración entre un sistema de gestión del conocimiento (*Knowledge Management Software*) y un framework de automatización de pruebas
- Uso de ontologías para mejorar la mantenibilidad de los scripts de automatización de pruebas
- Aplicación de la Arquitectura Dirigida por Modelos (MDA) enfocada a la creación de Pruebas Automáticas de Software Web

Bibliografía

Alégroth, E., Feldt, R. & Kolström, P. (2016). Maintenance of automated test suites in industry: an empirical study on visual GUI testing. *CoRR*, *abs/1602.01226*. Recuperado desde http://arxiv.org/abs/1602.01226

Amaricai, S. & Constantinescu, R. (2014). Designing a software test automation framework. *Informatica Economica*, *18*(1), 152-161.

Ary, J. (2013). *Instant drools starter*. Packt Publishing.

Black, R. (2009). *Managing the testing process: practical tools and techniques for managing hardware and software testing* (3rd). Wiley Publishing.

Burnstein, I. (2010). *Practical software testing: a process-oriented approach* (1st). Springer Publishing Company, Incorporated.

Cemus, K., Cerny, T. & Donahoo, M. J. (2015). Automated business rules transformation into a persistence layer. *Procedia Computer Science*, *62*, 312-318. doi:http://dx.doi.org/10.1016/j.procs.2015.08.391

d. Jesus, J. S. & d. Melo, A. C. V. (2015, julio). An architectural pattern to implement business rules in information systems. En *2015 ieee 17th conference on business informatics* (Vol. 2, pp. 80-87).

Galvis Lista, E. A. & González Zabala, M. P. (2014). Herramientas para la gestión de procesos de negocio y su relación con el ciclo de vida de los procesos de negocio: una revisión de literatura. *Ciencia e Ingeniería Neogranadina*, *24*(2), 37. Recuperado desde http://revistas.unimilitar.edu.co/index.php/rcin/article/view/392

Gojare, S., Joshi, R. & Gaigaware, D. (2015). Analysis and design of selenium webdriver automation testing framework. *Procedia Computer Science*, *50*, 341-346. doi:http://dx.doi.org/10.1016/j.procs.2015.04.038

Harmom, P. & Tregear, R. (2016). *Questioning bpm?* (1.ª ed.). Meghan-Kiffer Press.

International Software Testing Qualifications Board - ISTQB. (2016, enero). *Advanced level specialist syllabus in test automation – engineering*. Version draft beta, 13 Jan 2016.

Joy, J. & Singh, D. P. (2015, octubre). A generic framework design to enhance capabilities of an enterprise test automation framework. En *2015 international conference on applied and theoretical computing and communication technology (icatcct)* (pp. 207-212).

Karhu, K., Repo, T., Taipale, O. & Smolander, K. (2009, abril). Empirical observations on software testing automation. En *2009 international conference on software testing verification and validation* (pp. 201-209).

Loshin, D. (2013). Chapter 11 - business rules. En D. Loshin (Ed.), *Business intelligence (second edition)* (Second Edition, pp. 147-163). MK Series on Business Intelligence. Morgan Kaufmann. doi:http://dx.doi.org/10.1016/B978-0-12-385889-4.00011-9

McKinty, C. & Mottier, A. (2016). *Designing efficient bpm applications* (1.ª ed.). O'Reilly Media, Inc.

Myers, G. J. & Sandler, C. (2004). *The art of software testing*. John Wiley & Sons.

Rademakers, T. & Liempd, R. v. (2010). *Activiti in action: executable business process in bpmn 2.0* (1.ª ed.). Manning Publications Co.

Rosing, M. v., Scheer, A.-W. & Scheel, H. v. (2015). *The complete business process handbook: body of knowledge from process modeling to bpm* (1.ª ed.). Elsevier Inc.

Salatino, M., Aliverti, E. & De Maio, M. (2016). *Mastering jboss drools 6*. Packt Publishing.

Silver, B. (2006). Sketching Out BPM. *InfoWorld*, *28*(8), 26-35.

The Business Rules Group. (2000). *Defining business rules: what are they really?*

The Object Management Group Inc. (2010). *Business process model and notation (bpmn) specification*. OMG - The Object Management Group Inc.

von Halle, B. (2001). *Business rules applied: building better systems using the business rules approach* (1st). Wiley Publishing.

White, S. & Miers, D. (2008). *BPMN Modeling and Reference Guide*.

Witt, G. (2012). *Writing effective business rules*. Amsterdam: Elsevier (Morgan Kaufmann). Recuperado desde http://www.sciencedirect.com/science/book/9780123850515

Yu, W. & Patil, G. (2007). A workflow-based test automation framework for web based systems. En *Proceedings - ieee symposium on computers and communications* (12th IEEE International Symposium on Computers and Communications, ISCC '07, pp. 333-339). Computer Engineering Department, San Jose State University.

Zhu, X., Zhou, B., Li, J. & Gao, Q. (2008, julio). A test automation solution on gui functional test. En *2008 6th ieee international conference on industrial informatics* (pp. 1413-1418).

Referencias Web

Boyer, J. & Mili, H. (2011). Agile business rule development process, architecture, and jrules examples. Recuperado desde http://site.ebrary.com/id/10457854

EsperTech Inc. (2016). Faq - frecuent asked questions - esper. Recuperado el 6 de noviembre de 2016, desde http://www.espertech.com/esper/faq_esper.php

International Software Test Institute. (2016). Automated software testing. Recuperado el 7 de noviembre de 2016, desde http://www.test?institute.org/Automated_Software_Testing.php

Jboss. (2016a). Drools documentation: chapter 4. kie. Recuperado el 6 de noviembre de 2016, desde https://docs.jboss.org/drools/release/6.5.0.Final/drools-docs/html/ch04.html

Jboss. (2016b). Drools documentation: chapter 8. rule language reference. Recuperado el 23 de octubre de 2016, desde https://docs.jboss.org/drools/release/6.4.0.Final/drools-docs/html/ch08.html

WSO2 Inc. (2016). Wso2 business rules server documentation. Recuperado el 23 de octubre de 2016, desde https://docs.wso2.com/business-rules-server-documentation/WSO2+Business+Rules+Server+Documentation

ANEXOS

ANEXO A

Casos de Prueba del Software Caso de Estudio

A.1. Verificar departamentos disponibles en la vista de Clases Por Confirmar

Verificar departamentos disponibles en la vista de Clases Por Confirmar	
Propósito	Verificar que la totalidad de departamentos académicos se encuentren listados en los filtros de búsqueda de clases en la vista de Clases por Confirmar
Prerequisito	El usuario se encuentra autenticado
Datos de Prueba	▪ PeriodoAcademico = [1630] ▪ NoDepartamentos = [126]
Pasos	▪ Ingresar al link de menú ”Confirmación Clases” ▪ Ingresar a la opción desplegada en el submenu ”Confirmar Clases” ▪ Dar clic en el campo de lista desplegable de Departamentos ▪ Contar el no de departamentos
Resultados Esperados	▪ El periodo académico mostrado en pantalla sea igual a PeriodoAcademico ▪ El conteo del no de departamentos de la lista desplegable sea igual a NoDepartamentos

A.2. Verificar las clases *Sugeridas NO Evaluar*

Verificar las clases *Sugeridas NO Evaluar*	
Propósito	Verificar que un departamento contenga el número correcto de clases sugeridas a no evaluar, al igual que los nombres de dichas clases.
Prerequisito	El usuario se encuentra autenticado
Datos de Prueba	▪ *PeriodoAcademico* = [1630] ▪ *DepartamentoASeleccionar* = [Dpto Filosofía e Hist Derecho, Dpto Electrónica] ▪ *NoClasesNoConfirmadas* = [2, 277] ▪ *ClasesNoConfirmadas* = [[Tesis Doctoral], [Induc.a la Invest.Doctoral I, Practica Profesional, Práctica Profesional 2]]
Pasos	▪ Ingresar al link de menú "Confirmación Clases" ▪ Ingresar a la opción desplegada en el submenu "Confirmar Clases" ▪ Seleccionar el departamento definido en *DepartamentoASeleccionar* ▪ Dar clic en "Buscar"
Resultados Esperados	1. El número de clases que se encuentra en la pestaña de *Sugeridas NO Evaluar* es igual a *NoClasesNoConfirmadas* 2. En la lista de clases mostrada, incluye en la columna *Nombre Curso* los valroes de *ClasesNoConfirmadas*

A.3. Verificar las clases *Sugeridas para Evaluar*

Verificar las clases *Sugeridas para Evaluar*	
Propósito	Verificar que un departamento contenga el número correcto de clases sugeridas para evaluar, al igual que los nombres de dichas clases.
Prerequisito	El usuario se encuentra autenticado
Datos de Prueba	▪ *PeriodoAcademico* = [1630] ▪ *DepartamentoASeleccionar* = [Dpto Filosofía e Hist Derecho, Dpto Electrónica] ▪ *NoClasesConfirmadas* = [50, 101] ▪ *ClasesConfirmadas* = [[Circuitos Eléctricos, Circuitos en Frecuencia, Conversión de energía], [Argumentación Jurídica, Empresas y Derechos Humanos, Introducción al Derecho]]
Pasos	▪ Ingresar al link de menú "Confirmación Clases" ▪ Ingresar a la opción desplegada en el submenu "Confirmar Clases" ▪ Seleccionar el departamento definido en *DepartamentoASeleccionar* ▪ Dar clic en "Buscar"
Resultados Esperados	1. El número de clases que se encuentra en la pestaña de *Sugeridas NO Evaluar* es igual a *NoClasesNoConfirmadas* 2. En la lista de clases mostrada, incluye en la columna *Nombre Curso* los valroes de *ClasesNoConfirmadas*

A.4. Revisión Masiva de Clases

Revisión Masiva de Clases	
Propósito	Verificar que al activar la opción de ”Finalizar Revisión” en la pestaña de ”Sugeridas para Evaluar”, se realice la revisión masiva de las clases listadas.
Prerequisito	El usuario se encuentra autenticado, y ya ha realizado una búsqueda de las clases de un departamento específico en el módulo *Confirmar Clases*
Datos de Prueba	▪ *PersonaReviso* = ”Juan Camilo Salazar” ▪ *FechaRevision* = Fecha de ejecución de la prueba
Pasos	▪ Seleccionar la pestaña ”Sugeridas para Evaluar” ▪ Colocar el cursor sobre el botón ”Finalizar Revisión”, y verificar el texto de ayuda (si sale) ▪ Dar clic sobre el botón ”Finalizar Revisión” ▪ Volver a hacer clic sobre la pestaña ”Sugeridas para Evaluar”
Resultados Esperados	1. El texto de ayuda que sale al colocar el cursor sobre el botón ”Finalizar Revisión” contiene el texto *”Al dar clic, marcará las clases mostradas en pantalla como CONFIRMADAS y serán evaluadas por los estudiantes”* 2. Después de dar clic sobre el botón ”Finalizar Revisión” y regresar a la pestaña ”Sugeridas para Evaluar”, todas las filas del campo de ”Revisado Por” son iguales a *PersonaReviso*, y la fecha revisión es igual a *FechaRevision*

A.5. Desconfirmar una clase para evaluación

Desconfirmar una clase para evaluación	
Propósito	Verificar que se permita desconfirmar una clase individual, de la lista de Clases Sugeridas para Evaluar
Prerequisito	El usuario se encuentra autenticado, y ya ha realizado una búsqueda de las clases de un departamento específico en el módulo *Confirmar Clases*
Datos de Prueba	▪ NombreDepto: Dpto Literatura ▪ Clases: • CodCurso: 8280, CodClase: 5919, Docente: Quevedo Alvarado Maria Piedad • CodCurso: 8281, CodClase: 5922, Docente: Vaughan Caro Felipe ▪ PersonaReviso: Juan Camilo Salazar Rodriguez ▪ FechaRevision: Fecha de ejecucion de la prueba
Pasos	▪ Seleccionar la pestaña ”Sugeridas para Evaluar” ▪ Buscar la clase que esta en *Clase*, y dar clic en el botón con texto *Si* que se encuentra en la columna *Confirmado* ▪ Hacer clic en el botón ”Buscar”, y luego en la pestaña ”Sugeridas NO Evaluar”.
Resultados Esperados	1. Al dar clic en el botón con texto ”Si” de la clase respectiva, este cambia al texto ”No” 2. Luego de dar clic en el botón buscar, aparece la clase en la pestaña de ”Sugeridas NO Evaluar”.

ANEXO B

Casos de Uso del Software Caso de Estudio (SUT)

B.1. Diagrama de Casos de Uso del Software *Administrador Encuesta Docente*