UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

AUTOMATIZACIÓN DE PRUEBAS DE REGRESIÓN PARA REDUCCIÓN DE TIEMPO DE ENTREGA DE NUEVAS VERSIONES DE SOFTWARE

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN TECNOLOGÍAS DE LA INFORMACIÓN

ÁNGELA MARÍA CORTÉS PABÓN

PROFESOR GUÍA:
JOCELYN SIMMONDS WAGEMANN

MIEMBROS DE LA COMISIÓN:
JUAN ALVAREZ RUBIO
ALEXANDRE BERGEL
MARCELLO VISCONTI ZAMORA

SANTIAGO DE CHILE

2020

RESUMEN

Surecomp, empresa sobre la cual se desarrolló el presente proyecto es una compañía dedicada al desarrollo de productos software para el intercambio de información financiera, sus clientes son el sector bancario y empresas de comercio exterior.

La principal problemática identificada fue la inexistencia de un proceso formal para el área de calidad de software, que permitiera conocer las metodologías necesarias para la entrega de productos con estándares requeridos por el sector, esta incluye (1) la baja gestión de la documentación manejada por el equipo y el bajo conocimiento para el análisis de documentos técnicos presentados por otras áreas; (2) la extensión sobre los plazos dispuestos para la entrega de una nueva versión de software en cuanto a la ejecución de casos de pruebas.

Para dar solución a las problemáticas se propuso definir un proceso de calidad de software mediante una metodología cuantitativa-cualitativa, obteniendo como resultado 4 subprocesos para e área (1) planificación y organizar, (2) adquirir e implementar, (3) entregar y dar soporte (4) monitorear y evaluar; para cada uno de ellos se definió su descripción, artefactos de entradas/salidas y sus actividades.

Para dar respuesta la segunda problemática con la extensión sobre plazos de entrega, se genera la automatización de pruebas de calidad con la herramienta Selenium, en base a una arquitectura de clases propia para la empresa. De esta actividad se obtuvo como resultado la reducción del 60% a 37,42% del total del tiempo utilizado para la ejecución de pruebas de regresión.

Para conocer los resultados cualitativos del proyecto se realizar una encuesta realizada a los diferentes roles involucrados en el desarrollo del proyecto, en las respuestas a esta encuesta se reconoce gracias al proceso implementado, junto a las pruebas automatizadas la entrega de valor y productos de mejor calidad a los clientes; así como un orden establecido para las actividades del área.

Finalmente, gracias al desarrollo de este proyecto fue posible la apertura de un nuevo servicio de automatización que la empresa Surecomp, pudo implementar en sus sedes de Alemania e Israel.

DEDICATORIA

Dedico esta tesis a mis padres y hermano, que gracias a su apoyo incondicional pude realizar mis estudios en otro país y cumplir mi sueño de conocer nuevas culturas. A mis familiares y amigos que se encuentran en Colombia y en otras partes del mundo, con quienes gracias a las tecnologías hemos podido conectar para darnos mensajes de alegría y cariño. A todas las personas que he conocido en este hermoso país en el cual ya llevo más de 3 años, quienes me enseñaron una nueva forma de hablar español, y me abrieron siempre las puertas de sus casas.

Tabla de contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1: Introducción

1.1.Contexto

Actualmente, las empresas del sector financiero demandan una alta calidad sobre los productos software que soportan sus operaciones [1]. Esta exigencia requiere que las empresas desarrolladoras de software que proveen servicios a este sector adopten prácticas, metodologías o herramientas sobre sus procesos de aseguramiento de calidad y así disminuir los errores o fallos que pueden ser muy costosos para la cadena de valor de sus clientes.

Surecomp es una empresa dedicada al desarrollo de soluciones informáticas para el intercambio de información financiera, de tesorería y cadena de proveedores de la industria de bancos y empresas de comercio exterior. La compañía cuenta con 9 oficinas alrededor del mundo (Alemania, Argentina, Canadá, Chile, China, Estados Unidos, Israel, Reino Unido, Singapur), siendo la oficina chilena SULA – Lationamerica, la encargada de entregar servicios y mantenciones a clientes de América del Sur como BCI (Chile), BIE (Ecuador), ITAU (Chile – Paraguay), MONEX (México), entre otros.

Cada una de las soluciones que provee la empresa, debe cumplir con el estándar establecido por la Society for Worldwide Interbank Financial Telecommunication, quienes han establecido el standard "Swift" el cual provee un listado de mensajes para intercambio de información financiera, categorizados según el tipo de información que ha de ser comunicada [2].

Para el año 2018 y 2019 se ha publicado una actualización del estándar, de cumplimiento obligatorio [3]. La actualización comprende cambios en la mensajería, como por ejemplo, la creación de nuevos mensajes, especificación de nuevos campos por mensaje, entre otros. Por lo anterior, Surecomp ha planificado la actualización de cada una de sus aplicaciones para ajustarse a los cambios dispuesto por el standard Swift. SULA, es la encargada de realizar la mantención de 3 aplicativos instalados en cada uno de sus clientes: allTRA®, allNET® y Bancomex®, siendo allTRA el core principal sobre el envío y recepción de información bancaria, sobre el cual se enfocará la presente tesis. Este sistema es el encargado de la gestión de instrumentos de pago de comercio exterior como lo son: cartas de crédito, standby, garantías, entre otros, y cuyos usuarios finales en su mayoría son ingenieros comerciales especialistas de las áreas de comercio exterior de los bancos. El sistema permite emitir, enmendar, negociar y actualizar cada uno de los instrumentos de pago, enviando comunicaciones sobre estos eventos a cada uno de los interesados como lo son: clientes de comercio exterior, bancos nacionales e internacionales.

La empresa cuenta con un modelo de desarrollo de software Iterativo – Incremental [4], en donde cada versión del software contiene la integración de los desarrollos core (los cuales son transversales para todos los clientes), y ajustes de customización específica solicitada por algún cliente en particular. Los cambios de customización y desarrollo de baja complejidad son realizados por la sede que tenga asignada la mantención del banco solicitante; los ajustes y desarrollos de mayor complejidad son llevados a cabo por las sedes principales de Surecomp en Israel y Estados Unidos.

Para el cumplimiento de la actualización al estándar SWIFT, se ha planeado una nueva versión del sistema allTRA (Versión 2.8), sobre la cual el equipo de calidad deberá realizar las respectivas pruebas para cada uno de los clientes. El equipo de calidad está integrado por 3 personas: el jefe de área y 2 analistas QA. Las pruebas de software que el equipo de calidad lleva a cabo con la entrega de una nueva versión son las siguientes: pruebas funcionales, pruebas de regresión, y listas de chequeo de configuraciones e instalación.

1.2.Problema

En el ciclo de vida del desarrollo de software, una adecuada gestión del proceso de calidad permite el aseguramiento de la entrega de valor hacia los clientes. En el caso particular de la empresa Surecomp, esta actualmente cuenta con un proceso informal en el área de calidad de software, en donde los casos de prueba ejecutados son aquellos casos de negocio que han sido creados por el área de Business Consultant, quien es el área encargada del relacionamiento con los clientes y la definición del alcance de los cambios, adicionalmente crean la documentación necesaria que es entregada a la demás áreas sobre los requerimientos de mejora o ajuste que son solicitados por los clientes.

Sin embargo, se han detectado las siguientes problemáticas dentro del área de calidad:

- No existe un proceso de calidad formalizado, por lo tanto, no se cuenta con prácticas definidas que puedan ser ejecutadas y medidas.
- Actualmente, los casos de prueba se ejecutan de forma manual, lo cual requiere un alto uso de horas-hombre del equipo de testing en la ejecución de pruebas de regresión.
- No se gestionan los documentos de prueba (control de versionamiento, formalización de evidencias, entre otros), por lo que el conocimiento del proceso de testing se encuentra tácito en las personas que participan en el proceso.

Sobre este último punto, según mediciones realizadas por la Jefatura de Proyectos, área encargada del control y seguimiento de los proyectos llevados por la empresa; la ejecución de las pruebas de calidad de software de la versión 2.6 de los sistemas allTRA y allNET tomaron en promedio 1.823,25 horas-hombre por cliente, donde se estima que un 60% de este esfuerzo corresponde al sistema allTRA. Este proceso comprende 7 actividades (Ver Figura 1); de la cual se destaca la actividad de Ejecución de pruebas (QA Test Execution). En base a la estimación del 60%, esto significa que para la ejecución de pruebas de calidad de software para allTRA fueron necesarias aproximadamente 867,9 horas-hombres.

En el proceso de calidad se han definido 3 tipos de pruebas de calidad de software que son ejecutadas para la entrega de una nueva versión: pruebas funcionales, pruebas de regresión y listas de chequeo de configuraciones e instalación. Para estos tres tipos de pruebas, la jefatura del área de calidad ha estimado que el tiempo se distribuye de la siguiente forma: 60% para pruebas de regresión, 30% para pruebas funcionales y 10% para listas de chequeo de configuración y pruebas de instalación.

Activity	UPG QA allTRA - allNETT
Etiquetas de fila	**Suma de No of hours**
Documentation	237
Environment Setup & Maint	6,25
Management	25,5
Meetings & Reviews	73
QA Test Design	31,5
QA Test Execution	1446,5
QA Test Plan	3,5
Total general	**1823,25**

Figura 1. Control de horas – Proceso de calidad Versión 2.6 allTRA
Tomado de: STAR - Sistema para el control de horas de Surecomp

Actualmente, el sistema allTRA se encuentra implementado en 6 clientes en América del Sur: MONEX, ITAU CHILE, ITAU PARAGUAY, BIE, IMRPOSA, y CIBANCO. Con la anterior información, se ha estimado que para la ejecución de pruebas de calidad para la nueva versión 2.8 sería necesario un promedio de 867,9 horas-hombre por cliente, para un total de 5.207,4 horas-hombre. El tiempo que dispone el equipo de calidad son 7 meses desde la finalización del desarrollo core por parte de las otras áreas.

La empresa sólo cuenta con una capacidad de 60 horas-hombre mensuales por analista de QA; es decir, que el equipo de calidad en los 7 meses dispuestos para la ejecución de pruebas tiene una capacidad de 1.260 horas-hombre; siendo esta capacidad muy inferior al tiempo estimado total (5.207,4 horas-hombre). Esto, sumado a la falta de gestión de los casos de prueba, produce un nivel de incertidumbre en el proceso de certificación de la nueva versión de allTRA, y, por ende, genera incertidumbre acerca de la calidad del producto entregado finalmente a los clientes de Surecomp.

En síntesis, existen situaciones problemáticas que afectan el quehacer de la gestión de calidad de la empresa Surecomp, lo cual exigen la revisión e implantación de un adecuado proceso y herramientas acorde a sus necesidades.

1.3.Justificación del problema

Tomando como soporte la situación actual de la empresa, presentada en la sección anterior, las estimaciones realizadas por la jefatura de proyectos, sobre el tiempo necesario para completar la ejecución de pruebas de calidad para la versión 2.8 del sistema allTRA en cada uno de los clientes; que sobrepasa a la capacidad actual del área de calidad, hace necesario la implementación de un proceso formal que permitan agilizar el desarrollo de las tareas por parte del equipo.

El beneficio con el que se contará gracias a la implementación de un proceso formal para el área de calidad de Surecomp es lograr disponer de una visión integral de las principales actividades que

se deberían reconocer en un contexto de múltiples actividades interaccionadas de una organización [5] lo cual aporta rigurosidad y homogeneidad en el desarrollo de las pruebas. Este proceso será apoyado con herramientas de automatización de pruebas de calidad enfocado a las pruebas de regresión, las ventajas sobre la implementación de estas herramientas son [6]:

- La automatización de pruebas implica la generación y documentación de casos que aportan un repositorio de información.
- Se desvincula el factor humano sobre las pruebas, lo cual disminuye los factores de riesgos sobre posibles errores.
- La automatización de las pruebas, implica una disminución del costo de las mismas, no sólo porque se ejecutan rápidamente, sino además porque las podemos programar para que se ejecuten repetidamente y sin intervención humana.

Como desventaja sobre la implementación de casos de pruebas automatizados se reconoce el alto costo en el desarrollo de los casos, con factores como necesidad de ingeniero especialista con conocimientos en las herramientas; adicional del costo necesario para el mantenimiento de los casos y realizar actualización a los mismos.

En base a lo anterior, el presente Proyecto es de importancia para Surecomp ya que aporta estructura y formalidad en sus procesos; así como herramientas de soporte que permiten el cumplimiento de los tiempos y entrega de un sistema al cual se apliquen las pruebas de calidad necesarias, aportando valor sobre sus productos a los clientes.

1.4.Objetivos

Objetivo General

El objetivo de esta tesis es automatizar las pruebas de regresión que permita disminuir el esfuerzo de ejecución manual de estas pruebas. Para lograrlo se debe definir, formalizar e implementar un proceso de calidad de software. El foco de este trabajo es el sistema de comercio exterior allTRA versión 2.8.

Objetivos Específicos

1. Definir y validar un proceso formal de aseguramiento de calidad de software para la empresa.
2. Seleccionar y documentar los casos de pruebas de regresión a ser automatizados.
3. Codificar y estabilizar las pruebas de regresión seleccionadas, para apoyar el proceso de calidad de software definido en este trabajo de tesis.
4. Caracterizar el impacto de automatizar pruebas de regresión, orientado a la reducción de tiempos de ejecución de pruebas.

1.5.Metodología

La metodología seleccionada para el desarrollo del proyecto es el análisis cuantitativo y cualitativo en base a la metodología de automatización de pruebas. A continuación, se listan y detallan las actividades ejecutadas:

1. Referenciación y estado del arte: En esta primera fase se consolidará la base documental del proyecto, principalmente en términos conceptuales e investigativos. En términos generales, se profundizará sobre las temáticas propias del desarrollo del proyecto, procesos de calidad y automatización de pruebas. Como resultado de esta actividad se espera que el autor cuente con los fundamentos teóricos requeridos para la toma de decisiones y el desarrollo de las siguientes actividades.

2. Definición del proceso de calidad: En esta actividad se documentará el proceso actual del área de calidad. Se definirán las actividades y documentos estratégicos sobre los cuales deberá proceder el personal para las actividades de calidad de software. La formalización del proceso se realizará mediante una metodología iterativa-incremental revisando el impacto del proceso en conjunto con el personal que integra el área de calidad de la empresa.

3. Selección de los casos de prueba: En esta etapa se identificarán los casos de prueba de software a ser automatizados para lo cual se perfilarán los posibles casos utilizando una matriz que permita identificar los rasgos de las pruebas como lo son: complejidad, frecuencia de ejecución, automatizables o no.

4. Gestión de los casos de prueba: En esta actividad se documentarán los casos seleccionados, en una herramienta de gestión de pruebas de software que permita identificar el paso a paso de la prueba y los resultados esperados, así como el versionamiento y consolidación de evidencias sobre los planes de prueba.

5. Automatización de casos: Desarrollo de los scripts de automatización. En esta actividad se registrarán el esfuerzo que toma la generación de los scripts.

6. Ejecución de casos automatizados y toma de métricas: En esta actividad se ejecutarán los casos automatizados de software y se tomará las métricas necesarias para calcular el impacto del proyecto. En esta actividad se obtendrán como métricas el tiempo de ejecución de un caso automatizados vs. ejecución manual, tiempo completo de ejecución para todos los casos, cuantos pasos fallidos cuantos positivos.

7. Evaluación de la solución: En esta actividad se evaluarán los resultados del proceso. Mediante encuestas se evalúa el impacto obtenido de la implementación en la empresa.

1.6. Estructura del documento

En el capítulo 2 se presenta el marco teórico en el que se sustentan las bases teóricas del documento con estándares utilizados en el desarrollo de software, herramientas para la automatización de pruebas y el estado del arte con investigaciones e implementaciones símiles a la presente investigación.

Dentro del tercer capítulo, se presenta la situación actual de la empresa al momento del inicio del desarrollo del proyecto. El diseño del proceso, tareas, roles y artefactos son definidos en el capítulo 4.

A continuación, en el capítulo 5 se describe el sistema de casos automatizados en donde se seleccionan los casos de prueba, se definen los artefactos que deben ser implementados como documento de casos de pruebas, un modelo de clases y arquitectura para la implementación.

Para finalizar en el capítulo 6 se presentan los resultados sobre indicadores e impresiones del equipo de trabajo y el capítulo 7 con las conclusiones.

Capítulo 2: Marco teórico

En este capítulo se abordará en la sección 2.1 la definición de calidad de software, diferentes modelos del proceso de desarrollo de software que permiten a las empresas identificar como mejorar sus procesos para así entregar productos de calidad. Se especifican los tipos de pruebas que se permiten el aseguramiento de calidad y las herramientas que permiten la ejecución de estos tipos de pruebas, seguido por la sección 2.2 con el estado del arte en base a trabajos relacionados sobre la automatización de pruebas de software y definiciones relevantes para el desarrollo de este trabajo.

2.1.Conceptos fundamentales sobre la calidad de software

Para el desarrollo del presente proyecto es importante la comprensión del proceso de calidad como uno de los subprocesos que conforman el ciclo de vida del desarrollo de software. Para una mejor comprensión en la sección 2.1.1 se define el concepto de calidad de software y se describen algunos modelos que permiten evaluar el proceso de desarrollo en una empresa.

2.1.1 Calidad de software y modelos de evaluación del proceso de software

La definición de calidad sobre un producto software ha sido declarado por distintos entes, la norma ISO/IEC 25000 describe “La calidad del producto software se puede interpretar como el grado en que dicho producto satisface los requisitos de sus usuarios aportando de esta manera un valor. Son precisamente estos requisitos (funcionalidad, rendimiento, seguridad, mantenibilidad, etc.) los que se encuentran representados en el modelo de calidad, el cual categoriza la calidad del producto en características y subcaracterísticas.” [7].

La calidad de software es la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo documentados, y con las características implícitas que se espera de todo software desarrollado profesionalmente.” [8].

Para la empresa Surecomp la calidad de software está definida (empíricamente) como: el grado de cumplimiento entre los requisitos del cliente y las normas y estándares sobre los cuales han sido desarrollado sus sistemas.

Para el aseguramiento de la calidad en los productos de software, existen modelos que permiten definir o evaluar los procesos de desarrollo del producto. La principal finalidad de estos modelos de calidad del producto software es especificar y evaluar la calidad de los productos software, ya sea a través de medidas “internas”, directas de las propiedades inherentes del software o mediante medidas “externas”, indirectas del comportamiento del sistema del que forma parte. En la Figura 2, se detallan diferentes modelos de proceso que han sido utilizados por empresas de desarrollo de software, algunos de ellos para el modelamiento de proceso y otro para la evaluación del mismos [9].

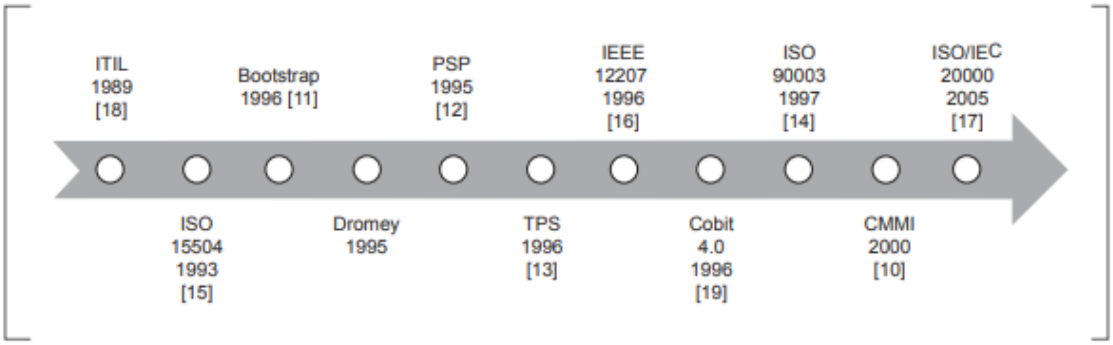


Figura 2. Línea de tiempo de proceso de software tomado de [9]

Uno de los modelos más utilizados en las empresas de construcción de software, con el propósito de verificar el cumplimiento de estándares de calidad a partir de la medición con niveles de madurez es CMMI (Capability Maturity Model Integration), el cual es relevante para la formalización de proceso presentada en las siguientes secciónes.

Este modelo se representa de dos maneras: escalonada y continua. El modelo escalonado está dirigido al software y permite clasificar las organizaciones en cinco tipos de nivel establecidos: Inicial, gestionado, definido, gestionado cuantitativamente y en optimización. Por su parte, el modelo continuo se enfoca al análisis de la capacidad de cada proceso inmerso en las áreas de la ingeniería de software y lo clasifica en uno de los siguientes seis niveles: Incompleto (0), ejecutado (1), gestionado (2), definido (3), cuantitativamente gestionado (4) y en optimización (5).

Las prácticas que mide del modelo CMMI son agrupadas según la relación en un área, que cuando se implementan de manera colectiva, satisfacen un conjunto de objetivos considerados imporantes para hacer mejorar esa áreas. Cada una de estas áreas de proceso incluye [10]:

- Objetivos, considerados importante para un ára de negocio. Por ejemplo, el área de proceso de gestión de requisitos contiene todas las prácticas relacionadas con la gestión de requisitos, el seguimiento de los cambios de requisitos, y el control de cambios.
- Prácticas.
- Material información.

2.1.2 Tipos de pruebas de calidad

Los tipos de pruebas de calidad son un grupo de actividades de prueba destinadas a probar características específicas de un sistema o parte de el, basado en objetivos de prueba específicos como son:

- Evaluar las características funcionales de calidad, tales como integridad, corrección y adecuación.
- Evaluar características de calidad no funcionales, como confiabilidad, eficiencia de rendimiento, seguridad, compatibilidad y usabilidad.

- Evaluar si la estructura o arquitectura del componente o sistema es correcta, completa, y según lo especificado.
- Evaluar los efectos de los cambios, como confirmar que se han reparado los defectos (confirmación de pruebas) y en busca de cambios involuntarios en el comportamiento resultantes del software o el entorno cambios (pruebas de regresión).

Los tipos de pruebas se dividen en: pruebas funcionales, pruebas no funcionales, pruebas de caja blanca, pruebas relacionadas a cambios [11], las que se describen a continuación.

a) Pruebas funcionales

Las pruebas funcionales de un sistema implican pruebas que evalúan las funciones que el sistema debe realizar. Los requisitos funcionales pueden describirse en productos de trabajo, como los requisitos comerciales, especificaciones, epopeyas, historias de usuarios, casos de uso o especificaciones funcionales.

La minuciosidad de las pruebas funcionales se puede medir a través de la cobertura funcional. La cobertura funcional es la medida en que las pruebas han ejercido cierta funcionalidad y se expresa como porcentaje del tipo de elemento cubierto. Por ejemplo, usando la trazabilidad entre pruebas y requisitos funcionales, el porcentaje de estos requisitos que se abordan mediante pruebas puede ser calculado potencialmente identificando brechas de cobertura.

El diseño y la ejecución de este tipo de pruebas puede implicar habilidades o conocimientos especiales sobre el modelo de negocio, en particular donde se implementa el software. Por ejemplo, pruebas funcionoales para un software de contabilidad que requieren conocimientos teóricos sobre nomenclaturas o formas de distribución contables.

b) Pruebas no funcionales

Las pruebas no funcionales de un sistema evalúan las características de los sistemas y el software, como la usabilidad, eficiencia de rendimiento o seguridad. La prueba no funcional es la prueba de "qué tan bien" se comporta el sistema. El descubrimiento tardío de defectos no funcionales puede ser extremadamente peligroso para el éxito de un proyecto.

La minuciosidad de las pruebas no funcionales se puede medir a través de una cobertura no funcional. La cobertura no funcional es la medida en que algún tipo de elemento no funcional ha sido ejercido por pruebas, y se expresa como un porcentaje del tipo de elemento que se cubre. Por ejemplo, usando trazabilidad entre pruebas y dispositivos compatibles para una aplicación móvil, el porcentaje de dispositivos que se abordan mediante pruebas de compatibilidad se pueden calcular identificando potencialmente brechas de cobertura.

c) Pruebas de caja blanca

Las pruebas de caja blanca derivan pruebas basadas en la estructura o implementación interna del sistema. La estructura puede incluir código, arquitectura, flujos de trabajo y / o flujos de datos dentro del sistema.

La minuciosidad de las pruebas de caja blanca se puede medir a través de la cobertura estructural. Cobertura estructural es la medida en que se ha ejercido algún tipo de elemento

estructural mediante pruebas, y se expresa como un porcentaje del tipo de elemento como condicionales de código, funciones con salidas, etc.

El diseño y la ejecución de la prueba de caja blanca pueden implicar habilidades o conocimientos especiales, como la forma en que el código fue construido, cómo se almacenan los datos (por ejemplo, para evaluar posibles consultas de la base de datos) y cómo usar las herramientas de cobertura y para interpretar correctamente sus resultados.

d) Pruebas de confirmación de cambios

Cuando se realizan cambios en un sistema, ya sea para corregir un defecto o debido a nuevos o cambios de funcionalidad, se deben realizar pruebas para confirmar que los cambios han corregido el defecto o implementó la funcionalidad correctamente y no causó ninguna consecuencia adversa imprevista. A continuación, se listan las pruebas de confirmación de cambios más conocidas:

- Prueba de confirmación: después de corregir un defecto, el software se puede probar con todos los casos de prueba que falló debido al defecto, que debe volver a ejecutarse en la nueva versión del software. El software también puede probarse con nuevas pruebas para cubrir los cambios necesarios para corregir el defecto. El propósito de una prueba de confirmación es confirmar si el defecto original ha sido corregido con éxito.
- Pruebas de regresión: es posible que se realice un cambio en una parte del código, ya sea una corrección u otro tipo de cambio, puede afectar accidentalmente el comportamiento de otras partes del código, ya sea dentro del mismo componente, en otros componentes del mismo sistema, o incluso en otros sistemas. Los cambios pueden incluir cambios en el entorno, como una nueva versión de un sistema operativo o sistema de gestión de bases de datos. Tales efectos secundarios no deseados se denominan regresiones. Las pruebas de regresión implican ejecutar pruebas para detectar tales efectos secundarios no deseados.
- Pruebas de aceptación de usuario: Son las pruebas formales con respecto a las necesidades del usuario, requerimientos y procesos de negocio, realizadas para determinar si un sistema satisface los criterios de aceptación que permitan que el usuario, cliente u otra entidad autorizada pueda determinar si acepta o no el sistema.

2.1.3 Herramientas para apoyar el proceso de calidad de software

En esta sección se detallan las herramientas que permiten el aseguramiento de la calidad de productos software en las diferentes etapas del ciclo de desarrollo en base a los tipos de pruebas presentados en la sección anterior y con el objetivo de definir posteriormente cuales de ellas son las adecuadas para ser implementadas en la empresa y dar apoyo al proceso propuesto como objetivo principal del presente proyecto.

a) Herramientas para gestión de pruebas

- Testlink: Es un sistema de gestión de pruebas de software basado en la web, de código abierto. Es gestionado por una comunidad de foristas y voluntarios que publican guías e información sobre cómo utilizarlo. Permite la sincronización entre los requerimientos de usuario y los casos de pruebas, donde se pueden crear proyectos que contienen casos de pruebas (ver Figura 3), para proyecto puede tener 1 o más planes de pruebas conformados por casos a los que se relacionan los compilados que han sido creados para una versión y los resultados de las ejecuciones.

  Es compatible con la ejecución automática y manual de casos. Además de esto, también admite la integración con muchos sistemas populares de seguimiento de defectos como Jira, Mantis, Bugzilla, Trac, etc. [12].

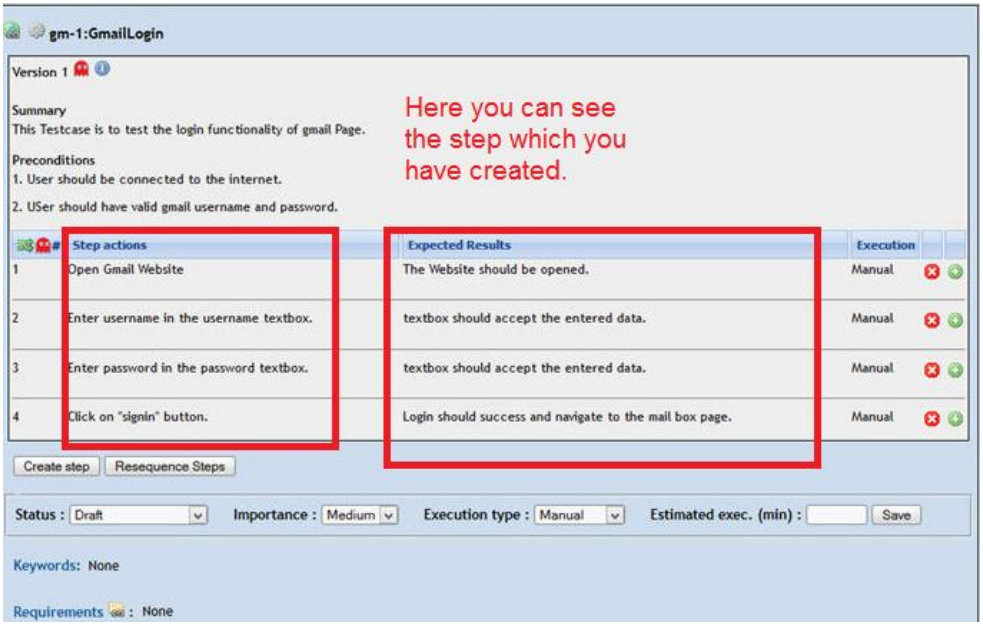


Figura 3. Ejemplo de caso de prueba en Testlink
Fuente: http://www.software-testing-tutorials-automation.com/2016/04/test-case-writing-in-test-link.html

- Herramientas de testing para Jira: Jira es un Sistema de seguimiento de proyectos e incidentes que permitre la instalación de plug-in(s) para la gestión de pruebas como Zephyr, Xray y Test Management. Es una herramienta web, y cada uno de sus plug-in son de pago. Permite la creación de casos de pruebas por producto, la creación de lanzamientos para control del versionamiento. Así como la relación entre casos de pruebas de tickets creados para su gestión.

- HP Quality center: Es una herramienta web de pago para la gestión de pruebas. El objetivo es el control de la calidad de software, incluyendo la gestión de requisitos, gestión de pruebas y los procesos de negocio para entornos y aplicaciones IT. Su escalabilidad permite la automatización de pruebas.

b) Herramientas para pruebas de carga y rendimiento

- Apache JMeter: Es una herramienta de código abierto para pruebas de carga que lleva a cabo simulaciones de request sobre cualquier recurso software. Inicialmente diseñada para pruebas de estrés en aplicaciones web, hoy en día, su arquitectura ha evolucionado no sólo para llevar a cabo pruebas en componentes habilitados en Internet (HTTP), sino además en Bases de Datos, programas en Perl, peticiones FTP y prácticamente cualquier otro medio. Mediante un interfaz es posible agregar cada uno de los pasos que componen las pruebas y configurar parámetros como tipo de petición, cantidad de repeticiones, entre otros como se muestra en la Figura 4.

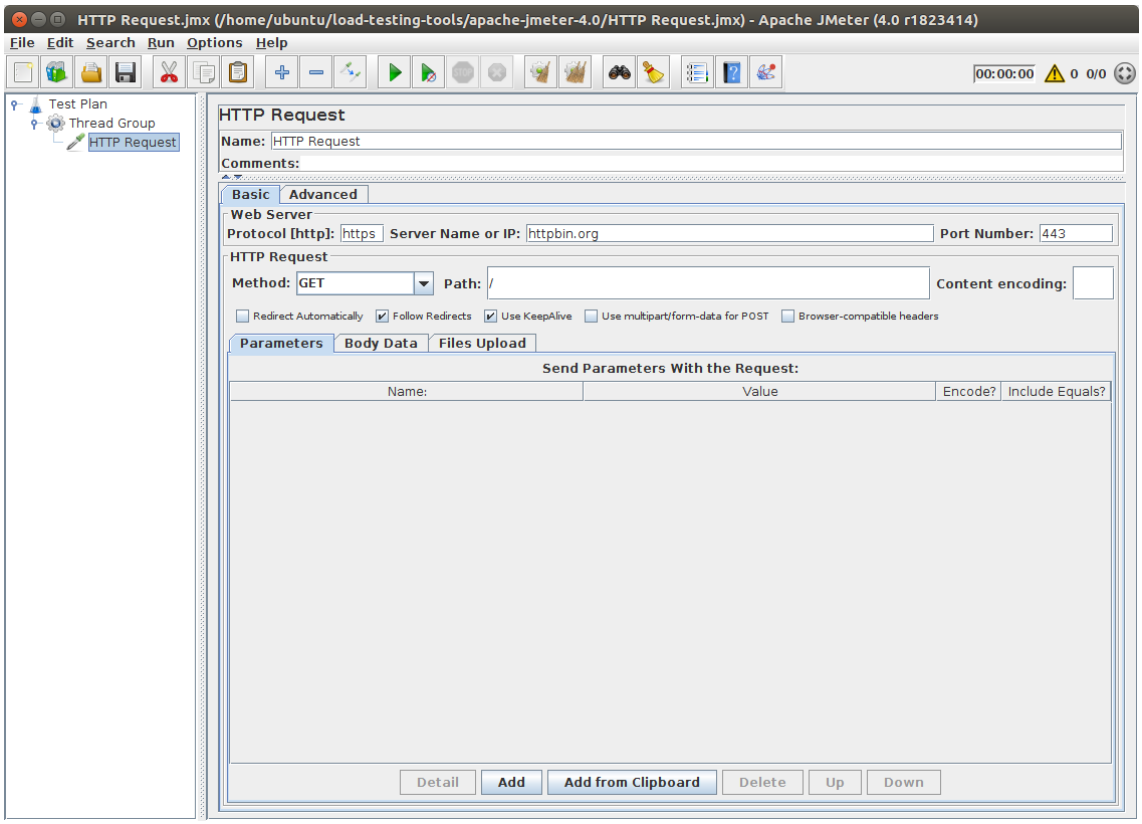


Figura 4. Apache JMeter Interface
Fuente: https://octoperf.com/blog/2018/03/29/jmeter-tutorial/

- FWPTT load testing: Es un programa de pruebas de carga de aplicaciones web en el cual, se puede grabar peticiones normales y Ajax. Funciona con aplicaciones desplegadas en asp.net, JSP, PHP, entre otros.

  Permite al usuario importar las sesiones de navegación realizadas con Fiddler. Estas solicitudes http se pueden importar y usar para generar una clase C# que llamará todas las solicitudes HTTP que el usuario haya registrado previamente. Permite grabar las acciones del usuario sobre su navegación utilizando cualquier navegador que le guste IE, Firefox, etc. La clase tiene funciones para manejar

los parámetros de consulta / publicación que puede usar para agregar una o más peticiones.

- Postman: Es una herramienta de creación y gestión de API(s) con entorno escritorio con extensiones para captura de peticiones en navegadores como Chrome y Mozilla. Tiene versiones gratuita y de pago que limitan la cantidad de API(s) gestionadas; permite la creación de pruebas testing para conocer peticiones y repuestas para validar el comportamiento de las mismas.

c) Herramientas para automatizar pruebas funcionales

- Selenium: Es un conjunto de herramientas de código abierto y gratuita que permite automatizar acciones que un usuario puede realizar sobre aplicaciones web. Cada herramienta dentro de este conjunto tiene un enfoque diferente para apoyar el proceso de automatización de pruebas. Los cuatro componentes son [13]:

  1. Selenium IDE: es un entorno de desarrollo integrado para scripts de Selenium. Su implementación es permitida solo como una extensión de Firefox o Google Chrome y permite grabar, editar y depurar pruebas. Este IDE incluye todo el Selenium Core, que permite grabar y reproducir de forma fácil las pruebas en el entorno real en el cual se ejecutarán. Se recomienda su uso para prototipos de pruebas, debido a que no es capaz de generar ciclos ni condiciones.

  2. Selenium RC (Remote Control): es una herramienta para automatizar pruebas de interfaz de usuario (UI) de aplicaciones web. Consta de dos componentes: a) un servidor que actúa como proxy para controlar e interactuar con un navegador web. b) bibliotecas para crear programas para el servidor usando una amplia gama de lenguajes de programación. Fue la herramienta original de Selenium para pruebas web pero a partir de la versión 2 fue integrado con WebDriver y se prefiere usar este último.

  3. Selenium WebDriver: también es una herramienta para automatizar pruebas UI de aplicaciones web pero implementa un enfoque más moderno y estable que Selenium RC. WebDriver, a diferencia de RC no utiliza un middleware sino que controla el navegador comunicándose directamente con él.

  4. Selenium Grid: se especializan en ejecutar múltiples pruebas a través de diferentes navegadores, sistemas operativo y máquinas. Puede conectarse con Selenium Remote especificando el navegador, la versión del navegador y el sistema operativo que desee. Hay dos elementos principales: hub y nodos.

- LeanFT: Solución de pruebas de software automatizada que incluye las siguientes características: Pruebas automatizadas en escenarios de pruebas

multicapa, incluyendo las pruebas de GUI y API, experiencia visual del usuario y el conjunto de herramientas potentes, prueba de tecnologías emergentes, con un innovador reconocimiento de objetos en UFT Insight, conversión sencilla de las pruebas manuales a las automatizadas, definición de un marco para una mejor gestión de las pruebas, gracias a la estrecha integración. Es necesario tener una licencia para su uso.

- SoapUI: Herramienta de código abierto gratuita desarrollada en JAVA que permite la ejecución de pruebas automatizadas funcionales, de regresión y de carga en diferentes para API(s) Web, orientada a serivicios (SOA) y transferencia de estados representacional (REST). Soporta múltiples protocolos como SOAP, REST, HTTP, JMS, AMF y JDBC; es ideal para aplicaciones construidas en base al llamado de servicios.

- UIPath: Es una herramienta de pago orientada a la automatización de procesos, en especial aquellos que involucran herramientas web o de escritorio. También permite la validación de resultados de ejecución para reconocer si la ejecución de un proceso fue lograda con éxito [15].

2.2.Estado del arte sobre automatización de casos de prueba

En esta sección se presenta y consolida la revisión de investigaciones e implementaciones realizadas por instituciones universitarias o empresas interesadas en la adopción de tecnologías para la ejecución de pruebas de software y asegurar la calidad de sus productos software.

Como elementos generales para la compilación de la información, se realizaron búsquedas en las siguientes bases de datos: IEEE y Science Direct; también en el buscador académico Google scholar; en base a las siguientes palabras claves: automation testing, software testing, quality assurance process; para un rango de tiempo de 2010 a 2019.

De los autores revisados, se identificaron diferentes puntos que convergen a las problemáticas presentadas como base del presente proyecto, algunos de ellos presentan las mismas ideas, entre las cuales se encuentran:

- El testing es crítico en aplicaciones web, esto para para reconocer que tan bueno es el funcionamiento de una aplicacón [16] [17].

- Los equipos de desarrollo cuyo proceso de pruebas era informal, evidenciaron mejoras en sus entregables, al ser estas integradas en etapas previas a la ejecución de pruebas y les permitió detectar posibles errores en etapas tempranas [17].

- El diseño de un mapa de procesos facilita la contextualización de las tareas a los equipos de desarrollo, y permite agruparla en conjuntos de actividades [5].

- Se debe realizar la automatización de pruebas de software para averiguar que tan bien funciona una aplicación, para evitar tareas tediosas, consumo de tiempo y reducir el factor humano de las pruebas manuales [15] [18].

- Para incrementar la calidad del producto, es necesario seleccionar casos de prueba mínimos que cubran todas las funcionalidades de prueba [18].

Asímismo se resaltan las siguientes ideas sobre la implementación de automatización para la ejecución de pruebas de software [16]:

- Es importante definir el propósito de iniciar un esfuerzo de automatización de las pruebas porque, aunque existen varias categorías de herramientas para hacerlo cada una tiene su propio propósito. Esto se hace relevante para el proyecto al momento de analizar los tipos de pruebas que se quieren llevar al sistema de automatización, ya que por su criticidad y recurrencia permitan generar un valor agregado a los productos [18].

- División modular de la arquitectura para reutilización de los casos de pruebas. Este punto fue tomado en cuenta para la definición de la arquitectura de clases del proyecto, y el cual permitió disminuir el tiempo de implementación entre casos de pruebas [19].

- Uno de los problemas identificados sobre las pruebas automatizadas es el hecho que la misma página podría mostrarse de maneras diferente en distintos navegadores y el contenido de la página. Esta problemática discutida por el auto referenciado se hizo evidente en la estabilización de los casos [17].

- Es muy importante realizar un análisis costo-beneficio sobre la implementación de las pruebas ya que en algunas experiencias estas no cumplen con las expectativas.

Sobre las investigaciones realizadas por los autores, se concluye que los esfuerzos de automatización requieren un proceso de ingeniería formal, definir lo que se automatizará (requisitos), diseñar la automatización, escribir probar e implementar los scripts, hacerle mantenimiento y definir su vida útil.

Esta información ha sido relevante para la establecer la metodología sobre la implementación de la solución.

Capítulo 3: Situación actual

El objetivo de este capítulo es establecer la situación de la empresa que se encuentra en sección 3.1. En la sección 3.2 se da a conocer los procesos, documentos y métricas del área de calidad que son ejecutados al momento de validar una nueva versión de allTRA previos al desarrollo del presente proyecto y en la sección 3.3 se describen los problemas identificados sobre la situación actual del área.

3.1.Situación actual de la empresa: Surecomp

Surecomp es una empresa fundada en Israel, proveedora de soluciones informáticas para intercambio de información financiera, de tesorería y cadenas de proveedores para bancos y empresas multinacionales. Actualmente, cuenta con 9 centros regionales de desarrollo y soporte en: Alemania, Argentina, Canadá, Chile, China, Estados Unidos, Israel, Reino Unido, Singapur; en las cuales se ha establecido una única gerencia.

Su centro regional en Chile se encuentra organizado por jefaturas, acordes a cada etapa del ciclo de vida del desarrollo de software, soportadas por áreas administrativas y jefatura de proyectos, las cuales son transversales a las actividades de la empresa. A continuación, se listan las jefaturas, su actividad principal y quienes la integran:

- Área de Bussines Consultant (BC), encargados del levantamiento y elicitación de los requerimientos de la mano con los clientes y primera línea de soporte. Integrada por un jefe de área y 2 consultores de negocio asignados a uno o más clientes.

- Área Técnica (TC) encargada de la administración de ambientes asignados a los clientes, así como actualización y gestión de configuración de los softwares. Se encuentra integrada por un jefe de área y 3 consultores técnicos.

- Área de Desarrollo e Investigación (R&D), encargada de los desarrollo y mejoras sobre los componentes de los diferente aplicativos. Esta área es liderada por un jefe de área quien no se encuentra presencialmente en el centro regional, 2 desarrolladores en remoto y 2 desarrolladores para el centro regional.

- Área de calidad (QA), encargada de la certificación de cambios en componentes software o de configuración, así como soporte a las pruebas de aceptación de usuario (UAT) realizadas por los clientes previo al paso de producción de una nueva versión de software. Esta área está integrada por 1 jefe y 2 analistas de calidad.

- Jefatura de proyectos, es el área encargada del seguimiento sobre el avance de los proyectos, así como la entrega de estimaciones de trabajo para los clientes.

AllTRA® es la solución de procesos back-office de Surecomp. Desarrollado en su totalidad en Java J2EE [14], cumple con los requisitos de financiamiento comercial de bancos; permite el procesamiento simplificado y automatizado de una amplia gama de transacciones de financiamiento comercial, incluyendo cartas de crédito comerciales y de reserva, bonos de ingresos

industriales (IRB), garantías, cobros, transferencias de fondos, reembolsos y pagos limpios. También brinda soporte mejorado para todas las formas de financiamiento, tales como aceptaciones, papel con descuento y préstamos a tasa fija, ajustable y variable.

En la Figura 5, se presenta la descripción de la arquitectura de la aplicación, la cual es posible desplegar en servidores Websphere y WebLogic [14]; siendo Internet Explorer el navegador autorizado a los clientes

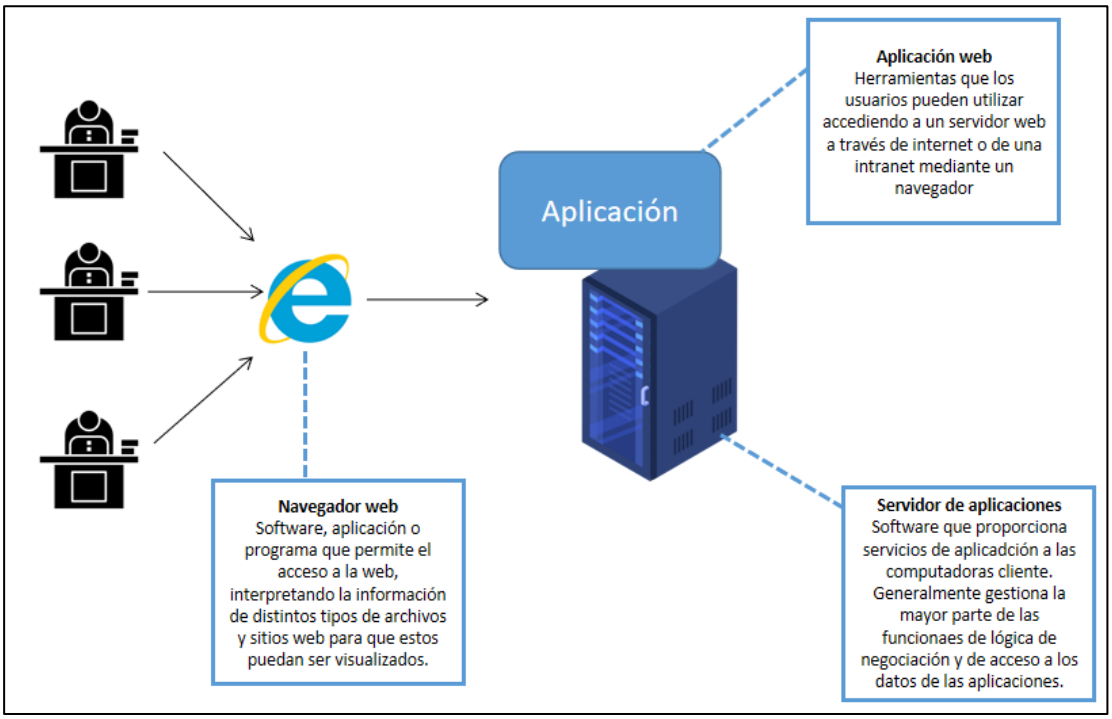


Figura 5. Descripción técnica de allTRA
Fuente: Surecomp

La creación de una nueva versión de software puede ser generada por diferentes razones: necesidad de una nueva funcionalidad por parte del cliente, cambios de tecnologías o cambios en las normativas que cumplen actualmente las funcionalidades del software. Siendo esta última la razón, por la cual Surecomp ha planeado la generación de la versión 2.8 del sistema allTRA como un proyecto de actualización.

A continuación, son detalladas (en orden cronológicos) las actividades que fueron realizadas para la versión 2.8 antes del desarrollo del presente proyecto y que se considera de relevancia mencionarlas para conocer la base sobre la que se desarrolló:

1. Publicación oficial de cambios en el standard Swift.
2. El jefe del área de R&D estableció, los desarrollos a realizarse dentro de la aplicación, obteniendo como resultado los siguientes documentos: documento de especificaciones funcionales (llamado FSD por sus siglas en ingles), información técnica de cambios “ITF”,

documento de cambios sobre interfaces de entrada llamado “Mapping changes” y documento de cambios para las interfaces de salida llamado “Templates changes”. El contenido de cada uno de estos documentos se describe a continuación:

- La información técnica de cambios es un documento - ITF (Word) que contiene los detalles técnicos y funcionales sobre los cambios que contiene una versión sobre los componentes de interfaz de usuario, reglas del negocio, base de datos, interfaces, entre otros.

- Mapping changes es un documento (Excel), en el cual se detallan los cambios sobre las interfaces de entrada del sistema, la información recibida y forma de almacenamiento en la base de datos para cada dato.

- Templates changes es un documento (Excel) en el cual se detallan los cambios o construcción de nuevas interfaces de salida del sistema, permite identificar en la construcción de las interfaces de donde es tomada la información sobre la base de datos.

3. En base a los documentos establecidos por el área de R&D, la jefatura de proyectos realizó una estimación de alto nivel en conjunto con los jefes de área. Se comienzan los desarrollos por parte de R&D.
4. Se realizó la primera entrega de la versión allTRA 2.8 por parte de R&D. Para este momento la principal documentación ha tenido diferentes cambios.
5. El área de BC comienza un análisis en base a la documentación de R&D y los documentos oficiales para reconocer el impacto funcional de los cambios sobre las configuraciones actuales para cada cliente. Como resultado se genera un documento de casos funcionales específicos para cada cliente.
6. El área de TC realizó un reconocimiento de los cambios sobre los componentes e interfaces; como resultado se genera el primer ambiente de pruebas general para la versión mencionado en el punto 4, en base a un cliente seleccionado por el área de R&D.
7. El área de QA recibe todos los documentos descritos anteriormente para su respectivo análisis y comienzo de las validaciones.

Las actividades listadas anteriormente son comunes y ejecutadas siempre que se realice el desarrollo de una nueva versión de software para todos los productos de la empresa

Para la estimación de tiempos se han tomado las horas de ejecución de pruebas de calidad de los 2 proyectos anteriores a la versión 2.8 de allTRA: actualización de versión 2.5 a 2.6 para el cliente ITAU Paraguay y actualización de versión 2.5 a 2.6 para el cliente MONEX.

La actualización de versión para el cliente ITAU Paraguay, fue un proyecto comprendido entre junio de 2014 y marzo de 2018 y en el cual se consumió un total de 4.561 horas en actividades de calidad, sobre un total de 22.477 de todas las actividades que fueron ejecutadas. En anexo 1 muestra un desglose de horas para este proyecto, por actividad, donde se destaca en amarillo las actividades correspondientes al área de calidad donde participaron 2 analistas de calidad.

La actualización para el cliente MONEX fue un proyecto desarrollado entre mayo 2017 a mayo 2018 y en el cual se consumió un total de 1.482,5 horas en actividades de calidad sobre un total de

6.654,25 de todas las actividades ejecutadas en el proyecto. El anexo 2 muestra un desglose de estas horas, por actividad.

De los anteriores datos se obtuvo que fue necesario un promedio de 1.371 horas-hombre en 12 meses para las actividades de calidad. En base a la fecha dispuesta por el estándar SWIFT para el uso del nuevo estándar, Surecomp cuenta con 7 meses para la ejecución de pruebas de calidad y entrega a los clientes de la versión 2.8 de allTRA, de los cual se estima necesarias 799 horas-hombre por proyecto y en total 4.794 horas-hombre para todos los clientes. La capacidad del equipo de calidad es de 4.284 para los 7 meses; de esto se ha llegado a la conclusión que la capacidad de trabajo el área no es suficiente, comparado con las horas estimadas para el aseguramiento de calidad de la versión 2.8 de allTRA.

Siendo calidad de software el tema principal de esta tesis, a continuación, se presenta en detalle en la sección área de calidad con sus procesos, capacidad de trabajo, ente otros.

3.2.Área de calidad

El área de calidad fue creada en el año 2016 para la sede de Surecomp en Chile, en donde anteriormente cada consultor de negocio era el encargado de realizar las validaciones sobre los desarrollos, ejecutando casos funcionales según sus conocimientos sobre los clientes asignados.

El área de Calidad está conformada por:

- Jefe de área: encargado de coordinar y velar por el cumplimiento de las actividades. Participa en reuniones de estimación, definición de nuevos proyectos y control de avance con áreas de otros centros regionales. También realiza actividades de los analistas de calidad. El jefe de área, quien anteriormente pertenecía al área de consultores de negocio no cuenta con conocimientos formales en tecnologías de la información en base a la experiencia sobre los aplicativos de Surecomp.

- Analistas de Calidad (2): encargados de la documentación y ejecución de casos de pruebas funcionales, pruebas de humo y de regresión. El área de conocimientos de uno de los analistas es comercio exterior y el segundo en ingniería de software.

Actualmente el área cuenta con un proceso informal, graficado en la Figura 6 el cual se detalla a continuación:

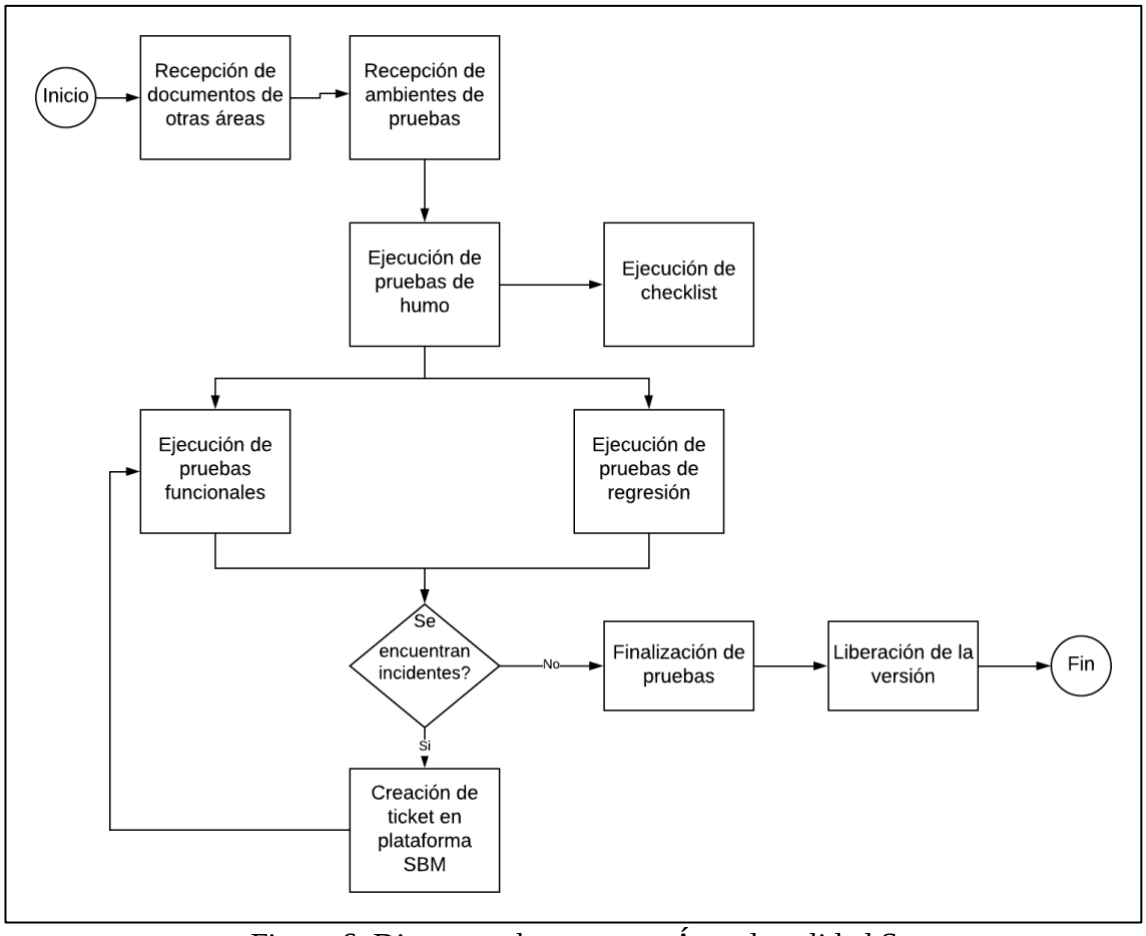


Figura 6. Diagrama de proceso - Área de calidad Surecomp
Fuente: Autor

Como insumo de entrada sobre el proceso, el área de calidad recibe el documento de especificaciones funcionales - FSD (pdf), el cual contiene al detalle de los requerimientos de cambio en el sistema que han sido pactados con los clientes y el documento de pruebas funcionales (Excel), el cual es desarrollado por el área BC y contiene los datos de pruebas que deben ser utilizados y los resultados esperados. Otro de los insumos es el ambiente general de pruebas, sobre el cual el área técnica realiza la instalación de los componentes de la nueva versión.

Sobre el ambiente general, los analistas de calidad ejecutan pruebas de humo. Estas pruebas son enfocadas en establecer una correcta respuesta entre allTRA y las interfaces de entrada y salidas que son ejecutadas 1 vez cada nuevo compilado de la versión. Al mismo tiempo se validan los ítems definidos en una lista de chequeo, con esto se establece que se hayan ejecutados procesos de base de datos y de actualización de la versión; en estas listas de chequeo se incluyen ítems como: validación de nuevos campos en la base de datos, validación de migración de datos, versión correcta, entre otros.

Una vez aprobada la lista de chequeo y las pruebas de humo se da inicio a la ejecución de pruebas funcionales y pruebas de regresión. Las pruebas funcionales, han sido definidas por parte del consultor de negocio, provienen de un documento Excel cuyo alcance son los cambios que afecten las principales funcionalidades del sistema y los cambios de configuración. La cantidad de pruebas es establecida según el conocimiento de cada consultor y son aproximadamente 50 casos por versión las cuales deben ser ejecutadas en cada uno de los clientes. En este documento se detallan

los valores que deben ser ingresados en cada caso de prueba, los valores de salida, y contiene un resumen principal para presentación el avance sobre la ejecución. Este formulario es propiedad del área de los consultores de negocio. La ejecución de estas pruebas se realiza de forma manual y como evidencia se adjuntan imágenes sobre las principales pantallas para cada caso en el documento Excel.

En paralelo a las pruebas funcionales son ejecutadas las pruebas de regresión, cuyo objetivo es establecer que los cambios de código no hayan afectado las principales funcionalidades del sistema. Para ello el equipo de analistas de calidad identifica un listado de pruebas sobre funcionalidad comunes que son identificadas como críticas (esto depende de la experticia y experiencia del analista), luego estos casos son ejecutados manualmente. Sobre estas pruebas se registran evidencias solo en caso de encontrar incidentes, los cuales son evidenciados mediante un ticket de cambio.

Si en alguna de las ejecuciones, funcionales o de regresión es identificada una inconsistencia; esta es registrada en un software de seguimiento y gestión de incidencias. El área de calidad genera un informe que resume la cantidad de incidencias encontradas para cada cliente, con el cual mediante un seguimiento diario se identifican los tickets que han sido resueltos por parte del área de desarrollo o aquellos que necesitan mayor información sobre lo reportado.

La totalidad de las pruebas funcionalidades y de regresión se debe realizar en 2 iteraciones, esta condición fue prestablecida por la jefatura del área de calidad. Cuando se finaliza la primera se espera nuevos compilados de la versión con la solución a una cantidad considerable de incidentes y es en ese momento que se comienza con la segunda ejecución. Con cada instalación de un nuevo compilado es necesario volver a ejecutar pruebas de regresión y así confirmar que los cambios no hayan afectado las anteriores funcionalidades. Una vez hayan sido corregidos todos los incidentes por parte del equipo de desarrollo y aprobada la solución por parte de los analistas de calidad se puede realizar la entrega formal de la versión al cliente. Es importante mencionar que para la empresa es más importante los plazos de entrega así que varias de las versiones son entregadas al cliente con errores conocidos (ticktes de cambio o incidentes sin resolver).

Una vez el cliente haya instalado la versión en sus ambientes de pruebas se da inicio al soporte de las pruebas de aceptación de usuario (ver sección 2.1.2) por parte de los analistas de calidad. Con el visto bueno de el cliente y el paso a producción de la versión por parte del cliente se da cierre al proyecto.

El área de calidad también es encargada de revisar los incidentes reportados por los clientes en sus ambientes de producción con prioridad crítico o alta y que requieren una pronta solución; este proceso no es detallado en este documento ya que está fuera del alcance del mismo.

En base a la situación actual de la empresa y el proceso del área de calidad detallado anteriormente; se identifican problemáticas que no permiten una validación ágil sobre los nuevos desarrollos de software.

3.3.Problemas identificados

El primer problema identificado, es la informalidad del proceso del área de calidad el cual ha sido empíricamente construido basado en los conocimientos y la experiencia del jefe de área.

Los documentos que dan inicio al proceso de validación por parte del área de calidad (documento de especificaciones funcionales y documento de pruebas funcionales), solo brindan detalles de los cambios sobre la capa de presentación y componentes funcionales; existen documentos generados por otras áreas que detallan técnicamente los cambios en módulos o componentes del sistema; lo que permitiría a los analistas de calidad identificar los tipos de pruebas que se deben realizar sobre una nueva versión; para esto es necesario capacitar a los analistas sobre como analizar la información contenida en estos documentos.

Tanto las pruebas funcionales como las de regresión son ejecutadas de forma manual, el área no cuenta con apoyo de herramientas claves que podrían agilizar las ejecuciones. Las pruebas de regresión se deben repetir con la instalación de cada compilado por lo que su ejecución manual requiere el consumo de más horas/hombre.

La definición de las pruebas de regresión y su ejecución depende del conocimiento de los integrantes del área, por lo tanto, al tener nuevos recursos no se les podrían asignar estas tareas. Adicional a ello, estas pruebas no cuentas con una evidencia o registro por lo que posteriormente se pueden encontrar inconsistencias con futuros cambios.

El área no cuenta con una base de documentación de pruebas, solo documentos informales sin versionamiento.

Capítulo 4: Diseño del proceso

En este capítulo se consolida el desarrollo sobre las tareas de la propuesta metodológica sobre el proceso. En la sección 4.1 se realiza una evaluación para reconocimiento de la madurez del proceso actual en el área de calidad se basó en el modelo CMMI [17], seguido por la sección 4.2 con el diseño del proceso versión y 4.3 segunda versión, finalizando con la evaluación de sus resultados en la sección 4.4.

4.1.Evaluación de madurez del proceso

Previo al diseño del proceso para el área de calidad, se aplicó un cuestionario basado en el modelo CMMI, desarrollado por la firma consultora Management Information Systems. Este cuestionario permite establecer el nivel control y ejecución de los sub-procesos y tareas que componen las actividades de un área de calidad de software. El cuestionario original en inglés, se divide en 20 sub-procesos con 173 preguntas; 20 preguntas sobre estrategias de pruebas, 7 sobre el modelo de ciclo de vida del software, 5 sobre puntos de involucramiento en el proceso de pruebas, 4 sobre estimación y planificación, 5 sobre técnicas de especificación de pruebas, 3 de técnicas de pruebas estáticas, 13 sobre métricas, 5 sobre herramientas de pruebas, 9 para ambientes de pruebas, 2 sobre ambientes de la compañía, 19 sobre el compromiso e involucramiento de la empresa con el proceso de calidad, 16 sobre función de las pruebas y capacitación, 8 sobre alcance y metodologías, 12 para medir la comunicación interna y externa al área, 8 preguntas de reporte de la información, 7 sobre gestión de defectos, 10 sobre gestión de "testware" – conocido como el conjunto de pruebas entre el que se encuentra las pruebas automatizadas _, 8 sobre la gestión del proceso de pruebas, 9 sobre evaluación de las estrategias de pruebas y para finalizar 11 preguntas sobre pruebas de bajo nivel.

La ejecución del cuestionario se realizó mediante una reunión en la cual participaron los integrantes del área de calidad (jefe y 2 analistas) y de la jefatura de proyectos (ingeniero del proyecto) para un total de 4 personas. La metodología de ejecución del cuestionario fue:

- La reunión fue liderada por un analista de calidad (autora del presente proyecto), quien en calidad de moderadora se encargó de presentar la metodología para la medición.
- Durante la evaluación, se leyeron en conjunto las afirmaciones o preguntas de cada ítem de los 20 sub-procesos (en inglés).
- El moderador fue el encargado de responder cualquier inquietud cuando alguno de los participantes no comprendía el nivel de las preguntas.
- Cada participante debía responder el nivel de control y ejecución que creía se tenía sobre el ítem evaluado por parte del área de calidad.
- El moderador completaba los valores dados por cada uno de los participantes sobre el documento Excel.
- Al finalizar la hoja de cálculo entrega el promedio de las respuestas para cada ítem, el nivel de control sobre cada sub-procesos y una serie de recomendaciones para el área.

Cada una de las preguntas debía ser respondida con un valor entre 0 – 10 en donde cada rango de valor correspondía a lo siguiente:

- 0-1: esta práctica no es necesaria y (casi) nunca se realiza.
- 2-3: esta práctica se requiere a veces, o se hace a veces.
- 4-5: esta práctica se requiere, pero no siempre se hace, o la práctica se realiza con regularidad, aunque no se requiere o verifica.
- 6-7: esta práctica es normalmente requerida y usualmente se hace.
- 8-9: esta práctica es obligatoria, se realiza y se verifica (la práctica se institucionalizada).
- 10: esta práctica está institucionalizada y es un ejemplo de clase mundial.
- ?: si el participante no sabe la respuesta
- NA: si la práctica no es aplicable.

En la Tabla 1, se presenta los resultados del cuestionario, para cada uno de los subprocesos se establece el promedio de puntos asignados; el cuestionario estabelece el nivel de control para cada uno de ellos donde, de 0 a 5 puntos el proceso es concocido como controlado lo que significa que las tareas de este proceso son ejecutadas manualmente sin importar el resultado de las mismas, de 5 a 7 en un proceso eficiente en donde la mayoría de las tareas que lo componen son ejecutadas y existen métricas definidas, los subproceso con un puntaje mayor a 7 son conocidos como optimizado en donde se cuenta con herramientas, metodologías o estrategías de ejecución, medición y mejora continua para cada una de las tareas. El cuestionario y las respuestas se encuentran en el Anexo 1, del presente documento.

No.	Sub-proceso	Promedio respuesta	Nivel de gestión para el sub-proceso
1	Estratégia de pruebas	3,02	controlado
2	Modelo de ciclo de vida	4,75	controlado
3	Puntos de involucramiento	5	controlado
4	Estimación y planificación	5,75	eficiente
5	Tecnicas de especificación de pruebas	4,33	controlado
6	Tecnicas de pruebas estáticas	6	eficiente
7	Metricas	7,25	optimizado
8	Herramientas de pruebas	0,83	controlado

Tabla 1. Resultados evaluación de madurez, continua en Tabla 2
Fuente: Autor

No.	Sub-proceso	Promedio respuesta	Nivel de gestión para el sub-proceso
9	Ambientes de pruebas	6,86	eficiente
10	Ambiente de la oficina	8	optimizado
11	Compromiso y motivación	5,94	eficiente
12	Funciones de las pruebas y capacitación	6,41	eficiente
13	Metodología de alcance	6,38	eficiente
14	Comunicación	5,06	controlado
15	Reportes	6,43	eficiente
16	Gestión de defectos	6,83	eficiente
17	Gestión de “testware”	5,11	controlado
18	Gestión del proceso de pruebas	5,88	eficiente
19	Evaluación	4,625	Controlado
20	Pruebas de bajo nivel	4,6	Controlado

Tabla 2. Continuación tabla 1.
Fuente: Autor

Al analizar los resultados se logra determinar que de los 20 subprocesos evaluados 9 de ellos con controlados, 9 eficientes y 2 de ellos optimizados. Los 3 subprocesos con menor puntaje son: herramientas de prueba, estrategias de pruebas y técnicas de pruebas; siendo este resultado coherente a la problemática expuesta en la sección 3.3. Gracias a esta evaluación se definen estos 3 subprocesos como el foro de evaluación y formalización para el proceso del área de calidad.

4.2.Diseño del proceso (Versión I) y su validación

Uno de los objetivos planteados para solucionar las problemáticas identificadas en la situación actual es la definición de un proceso formal para el área de calidad de la empresa.

En la Figura 7, se detalla la primera versión diagramada para el proceso en base al proceso informal descrito en la sección 3.2; en color verde se han identificado nuevas etapas o que tienen cambios en su definición.

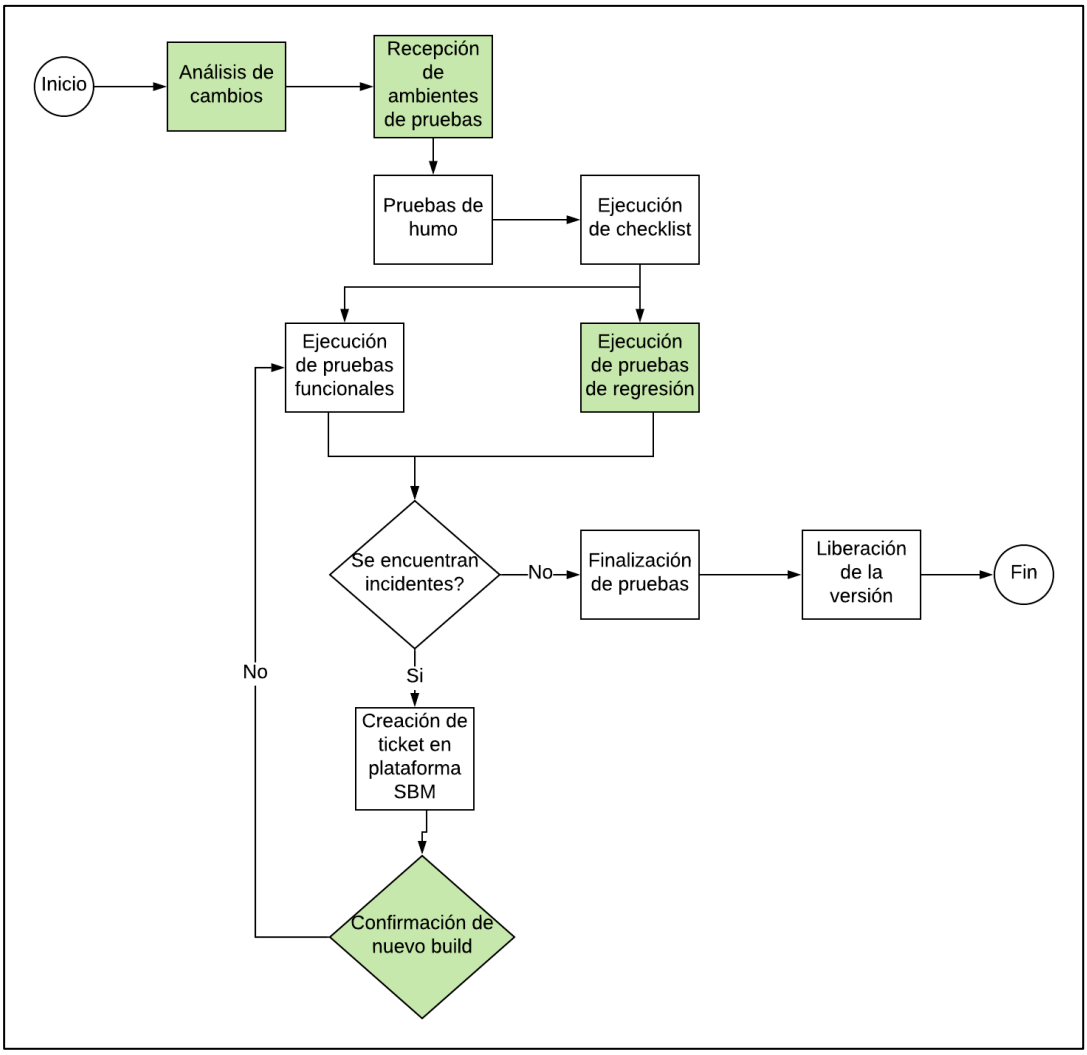


Figura 7. Versión 1 del proceso propuesto para el área de calidad

Como insumo de entrada sobre el proceso, para la etapa de Análisis de cambios se propone activar el uso de los documentos tanto funcionales como técnicos para el análisis sobre la cobertura de las pruebas que se deben ejecutar para la nueva versión. En total se utilizarían los siguientes documentos: documento de especificaciones funcionales – FSD, documento de pruebas funcionales, documento oficial de cambios, información técnica de cambios – ITF, mapping changes, template changes, los cuales fueron descritos en la sección 3.1, y su relevancia es dar a conocer los componentes y alcances de los cambios que realizará el área de desarrollo y así establecer la cobertura de las pruebas.

Cabe mencionar que cada uno de estos documentos tiene diferentes versiones las cuales son liberadas con el avance del desarrollo del proyecto.

El cambio propuesto para la etapa de Recepción de ambientes es el paso del control de este proceso del área de desarrollo de software al área de calidad, quien debe solicitar la creación de los ambientes que sean necesarios para las pruebas y gestionar los momentos sobre los cuales se podrán realizar actualizaciones sobre estos ambientes.

Para la etapa de Ejecución de pruebas de regresión se propone primero el análisis de los casos de prueba necesarios para cubrir este tipo de pruebas dentro del sistema. Estos casos deben ser documentados, adicional a la construcción de un sistema de ejecución de pruebas automatizadas.

Se agrega una nueva etapa dentro del proceso: Confirmación de nuevo compilado, en donde se comprueba si existe un nuevo compilado liberado por el área de R&D, esto permite una ágil instalación de los componentes, actualización de las mejoras, rápida solución a los problemas identificados en las primeras revisiones y ejecutar continuamente las pruebas de regresión automatizadas. Esta nueva etapa fue posible agregarla gracias a la generación de nueva documentación por parte del área técnica que incluye al detalle los cambios de cada instalación y los tickets solucionados de la plataforma SBM. La instalación del nuevo compilado es solicitada por el Área de Calidad lo que permite un mayor control de cambios.

Una vez realizada la formalización de la propuesta, se validó este proceso aplicándolo en el proyecto de actualización a la versión 2.8 de allTRA. Esta evaluación se realizó al finalizar el primer mes de implementación de la propuesta, por medio de una reunión de retrospectiva en donde participaron los integrantes de el equipo de calidad. De este hito se obtuvo la siguiente retroalimentación:

- Se hace necesaria la división del proceso del área en subprocesos los cuales son ejecutados en diferentes momentos según el avance de los proyectos.

- Se debe definir en un mayor detalle las tareas y/o actividades que componen el subproceso de automatización de pruebas, para que sea un proceso formal desde el inicio de las actividades.

- Es importante la definicion de la gobernabilidad dentro del área, lo que deberá incluir la definición de roles, artefactos de entradas / salida, y actividades.

4.3.Versión definitiva del proceso

La segunda versión del proceso del área de calidad se basa en la guía de mejores prácticas presentadas en el framework COBIT [18]. Para el desarrollo del presente proyecto se ha elegido la orientación a procesos del modelo, cuya estructura se divide en dominios y procesos. Los dominios son agrupaciones de procesos que corresponden a una responsabilidad y los procesos son una serie de actividades controlables con objetivos claros y un resultado medible.

COBIT define las actividades de TI en un modelo genérico de procesos organizado en cuatro dominios: planear y Organizar (PO), Adquirir e Implementar (AI), Entregar y Dar Soporte (DS) y Monitorear y Evaluar (ME). Los dominios se equipararán a las áreas tradicionales de TI de planear, construir, ejecutar y monitorear.

En la siguiente sección se detalla la definición de cada uno de los dominios, siendo cada uno de ellos orientados a los objetivos del área de calidad.

4.3.1 Definición de dominios

El primer de los dominios presentados por el marco de trabajo es: Planear y organizar. Este dominio cubre las estrategias y las tácticas, y tiene que ver con identificar la manera en que TI puede contribuir de la mejor manera al logro de los objetivos del negocio. Además, la realización de la visión estratégica requiere ser planeada, comunicada y administrada desde diferentes perspectivas. Finalmente, se debe implementar una estructura organizacional y una estructura tecnológica apropiada. Para este dominio se deben responder las siguientes aristas:

- ¿Están alineadas las estrategias de TI y del negocio?
- ¿La empresa está alcanzando un uso óptimo de sus recursos?
- ¿Entienden todas las personas dentro de la organización los objetivos de TI?
- Se entienden y administran los riesgos de TI?
- ¿Es apropiada la calidad de los sistemas de TI para las necesidades del negocio?

Como resultado a esta definición se han definido 5 procesos que se alinean al dominio, correspondiendo al conjunto de tareas que se llevan a cabo previo a la recepción y una nueva versión de software y/o ajustes del mismo. Los procesos que componen este dominio son:

1. Definición de alcance del proyecto.
2. Recepción y análisis de documentación.
3. Administración de ambientes.
4. Definición de tipo de pruebas.
5. Estimación de esfuerzo.

El segundo de los dominios del marco de trabajo es: Adquirir e Implementar, para llevar a cabo la estrategia de TI, las soluciones de TI necesitan ser identificadas, desarrolladas o adquiridas así como implementadas e integradas en los procesos del negocio. Además, el cambio y el mantenimiento de los sistemas existentes está cubierto por este dominio para garantizar que las soluciones sigan satisfaciendo los objetivos del negocio. Este dominio, por lo general, cubre los siguientes cuestionamientos:

- ¿Es probable que los nuevos proyectos generan soluciones que satisfagan las necesidades del negocio?
- ¿Es probable que los nuevos proyectos sean entregados a tiempo y dentro del presupuesto?
- ¿Trabajarán adecuadamente los nuevos sistemas una vez sean implementados?
- ¿Los cambios no afectarán a las operaciones actuales del negocio?

Los procesos definidos para este dominio son:

1. Definición de cobertura de casos de pruebas.
2. Especificación y documentación de casos de pruebas.
3. Ejecución de casos de pruebas
4. Gestión de incidencias
5. Automatización de pruebas

A continuación se desarrolla del tercero de los dominios: Entregar y dar soporte, el cual cubre la entrega en sí de los servicios requeridos, lo que incluye la prestación del servicio, la administración de la seguridad y de la continuidad, el soporte del servicio a los usuarios, la administración de los datos y de las instalaciones operativos. Por lo general responde a las siguientes preguntas:

- ¿Se están entregando los servicios de TI de acuerdo con las prioridades del negocio?
- ¿Están optimizados los costos de TI?
- ¿Es capaz la fuerza de trabajo de utilizar los sistemas de TI de manera productiva y segura?
- ¿Están implantadas de forma adecuada la confidencialidad, la integridad y la disponibilidad?

Para dar respuesta a las preguntas de este dominio se establecen los siguientes procesos:

1. Liberación de versiones
2. Soporte a pruebas UAT

Monitorear y evaluar es el último de los dominios, al cual define que todos los procesos de TI deben evaluarse de forma regular en el tiempo en cuanto a su calidad y cumplimiento de los requerimientos de control. Este dominio abarca la administración del desempeño, el monitoreo del control interno, el cumplimiento regulatorio y la aplicación del gobierno. Por lo general abarca las siguientes preguntas:

- ¿Se mide el desempeño de TI para detectar los problemas antes de que sea demasiado tarde?
- ¿La Gerencia garantiza que los controles internos son efectivos y eficientes?
- ¿Puede vincularse el desempeño de lo que TI ha realizado con las metas del negocio?
- ¿Se miden y reportan los riesgos, el control, el cumplimiento y el desempeño?

Para el área de calidad se han definido 2 procesos que permiten dar soporte a este último dominio:

1. Evaluación de procesos
2. Métricas

4.3.2 Diagrama de flujo de los procesos

En la Figura 8, se presentan los diagramas de flujo para los procesos de cada dominio. Los 3 primeros dominios son presentados de forma vertical ya que su ejecución es secuencial a diferencia del dominio de monitorear y evaluar que puede ser ejecutado en cualquier punto o estado del área o proyecto

.

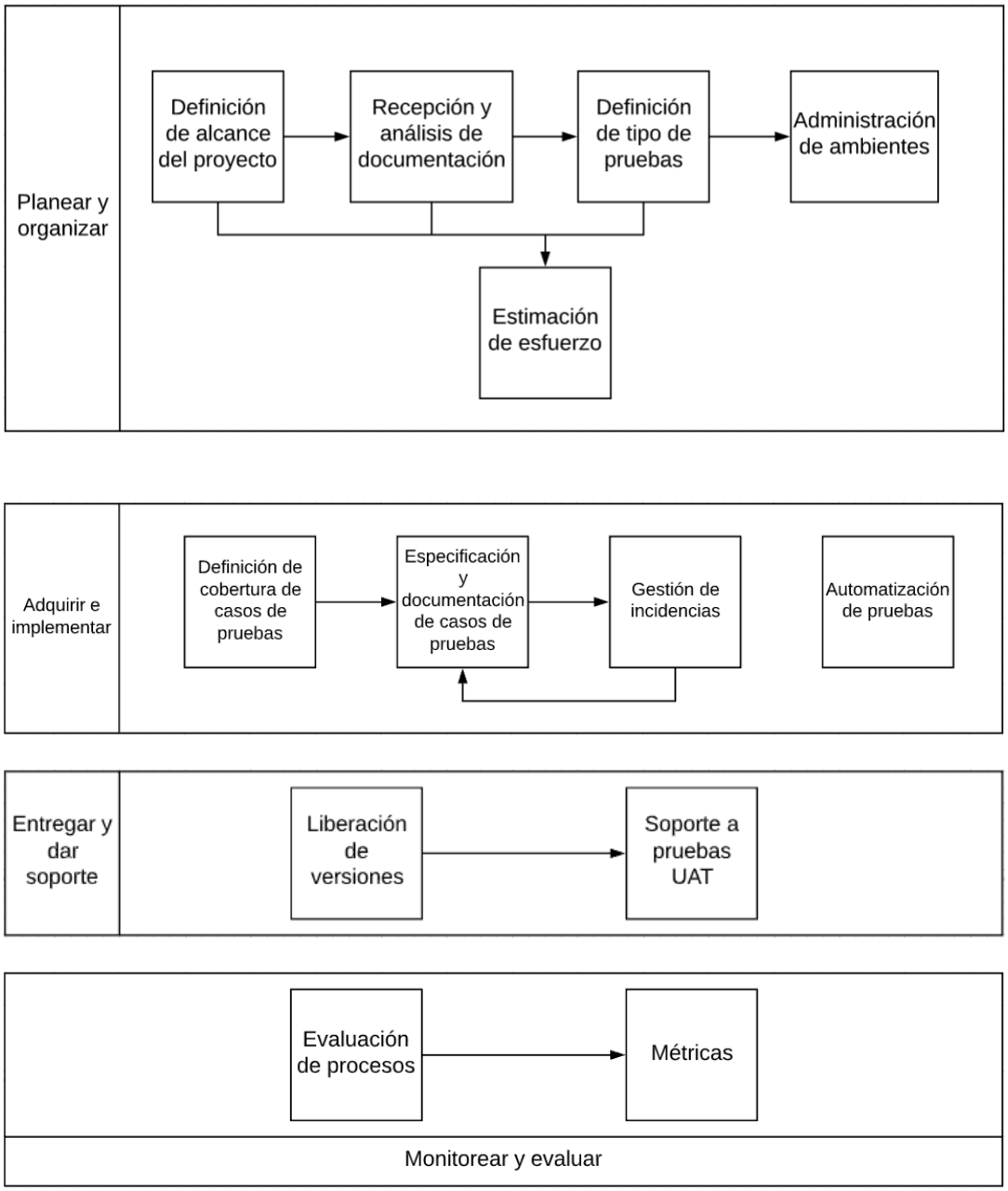


Figura 8. Procesos de los dominios del área de calidad
Fuente: Autor

4.3.3 Detalle de los procesos

De cada uno de los procesos que componen los dominios definidos para el área se presenta la definición de cada uno, en donde se detalla: descripción, artefactos de entrada / salida, acción o tareas ejecutadas sobre cada uno y los roles involucrados. A continuación en la Tabla 3, se presenta un ejemplo para el proceso de repción de documentación el dominio de planear y organizar. Los otros procesos se encuentran en el Anexo 4.

<table>
<tr><th colspan="3">Proceso: Recepción de documentación</th><th>Dominio: Planear y organizar</th></tr>
<tr><td>Descripción del proceso</td><td colspan="3">En este proceso se realiza la recepción y análiss de la documentación por parte de las áreas que integran el desarrollo de software (estos documentos puede provenir de otras sedes de la empresa)</td></tr>
<tr><td>Entradas</td><td>- Documento de especificaciones funcionales – FSD.
- Documento de pruebas funcionales.
- Documento oficial de cambios, información técnica de cambios – ITF
- Mapping changes.
- Template changes.</td><td>Salidas</td><td>Presentación de cambios</td></tr>
<tr><td>Acciones</td><td colspan="3">- Solicitar los documentos a cada una de las áreas correspondientes.
- Organizar la documentación en el repositorio correspondiente.
-Generar presentación de cambios y socializar con los integrantes del área de calidad y el jefe de proyectos asignado.
- Definir los tipos de pruebas de alto nivel que se debería ejecutar sobre los cambios presentados.</td></tr>
<tr><td>Roles</td><td colspan="3">Jefe calidad, analistas de calidad, jefe de proyectos.</td></tr>
</table>

Tabla 3. Proceso recepción de documentación
Fuente: Autor

4.4.Validación de la versión definitiva del proceso

Una vez finalizada la definición de los procesos para el área de calidad, estos fueron ejecutados sobre el proyecto allTRA 2.8, para la validación de resultados se llevó a cabo por segunda vez el cuestionario de diagnóstico de madurez que se detalla en la sección 4.1, para lo cual se utilizó la misma metodología presentada en esa sección. En este caso solo se contó con la participación del jefe de área de calidad y analista del área quien lideró el desarrollo del presente proyecto.

Como resultado para cada subproceso del área se obtuvieron los resultados presentados en la Tabla 4, 19 de los 20 subprocesos evaluados tuvieron una mejora en comparación a la primera evaluación.

No.	Sub-proceso		Nivel de gestión para el sub-proceso
1	Estratégia de pruebas	8	optimizado
2	Modelo de ciclo de vida	7	eficiente
3	Puntos de involucramiento	7	eficiente
4	Estimación y planificación	8	eficiente
5	Tecnicas de especificación de pruebas	8	optimizado
6	Tecnicas de pruebas estáticas	8	optimizado
7	Metricas	9	optimizado
8	Herramientas de pruebas	7	eficiente
9	Ambientes de pruebas	9	optimizado
10	Ambiente de la oficina	8	optimizado
11	Compromiso y motivación	7	eficiente
12	Funciones de las pruebas y capacitación	8	optimizado
13	Metodología de alcance	8	optimizado
14	Comunicación	8	optimizado
15	Reportes	8	optimizado
16	Gestión de defectos	9	optimizado
17	Gestión de “testware”	7	eficiente
18	Gestión del proceso de pruebas	8	optimizado
19	Evaluación	7	eficiente
20	Pruebas de bajo nivel	6	eficiente

Tabla 4. Resultados implementación proceso definitivo para el área de calidad

Capítulo 5: Descripción del sistema de ejecución de pruebas de software automatizadas

En este capítulo se desglosan las tareas llevadas a cabo para la implementación del sistema de ejecución de pruebas automatizadas. En la sección 5.1 se detalla las actividades realizadas para la definición de la tecnología a ser utilizada; sección 5.2 la definición y documentación de los casos de pruebas; 5.3 descripción de la arquitectura de despliegue y de clases, en la sección 5.4 el desarrollo de los scripts de pruebas; en la última sección se presenta la estabilización, ejecución y evidencia de las pruebas.

Para dar soporte a la ejecución de pruebas para la versión 2.8 de allTRA, como parte de esta tesis se construyó un sistema para la ejecución de pruebas automatizadas al cual se implementaron pruebas de regresión ya que son las pruebas de mayor repetición en los proyectos de software y deben ser ejecutados con cada nuevo compilado del producto. Este proyecto fue liderado por la autora del presente proyecto como analista de calidad y fue integrado un desarrollador del área de R&D. La construcción del sistema se estableció en 5 fases: Definición de tecnologías, documentación de los casos, definición de arquitectura, creación de los scripts y ejecución de las pruebas.

Como primera tarea se reconocen las características técnicas del sistema allTRA (ver sección **Error! Reference source not found.**) siendo las más relevantes: es un sistema web que se despliega en los servidores de aplicaciones WebSphere y Weblogic, con motores de base datos SqlServer y Oracle Database y cuyo único browser certificado es Internet Explorer.

5.1.Definición de tecnología

Para la definición de la tecnología a ser utilizada para la automatización de las pruebas se eligieron 3 entornos de pruebas, elegidas al soportar la automatización, así como tener mayor información sobre su implementación en la web, foros, videos, entre otros. Sobre cada una de las herramientas se realizó una prueba de factibilidad para determinar su comportamiento con el sistema allTRA y determinar el nivel de dificultad de su implementación. De las herramientas presentadas en la sección 2.1.3 se determinó evaluar: Selenium, LeanFT, UIPath.

La prueba de factibilidad se realizó sobre un caso de pruebas básico identificado para el sistema allTRA; el cual comprendía, en resumen: ingreso al sistema, selección de funcionalidad, llenar campos mandatorios, validación de resultados y cierre de funcionalidad.

En la Tabla 5, se resumen las principales características evaluadas en cada herramienta:

Herramienta / Característica	Selenium	LeanFT	UIPath
Código Abierto (Si / No)	SI	No	No
Es de fácil aprendizaje	No	No	Si
¿Se encuentra documentación en la web?	Poca	Poca	Poca
¿Tiene interfaz de usuario?	No	No	Si
Browsers aplicables	Todos [1]	Todos	Todos

Tabla 5. Comparativa herramientas de automatización
Fuente: Autor

Como resultado de la prueba de concepto se encuentra lo siguiente:

- Selenium: Para las pruebas con esta herramienta se realizó la descarga del componente webdriver, y el driver correspondiente a internet explorer desde la página oficial; posterior a ello fue integrado el driver con código JAVA desde eclipse como interfaz de desarrollo. Para la resolución de dudas se realizaron búsquedas en video tutoriales en youtube en donde se consigue la mayoría de documentación sobre la herramienta, así como búsqueda de respuestas en stackoverflow. Se identificó como necesario que el desarrollador de pruebas automatizadas debe tener conocimientos previos en html 5 para entender como definir al controlador de la página web que acceda y realice acciones sobre los objetos de la interfaz. Como resultado fue posible concluir el caso de pruebas.

- LeanFT: Para la prueba de factibilidad con esta herramienta se hizo necesario descargar un ejecutable demo, el cual permite integración a eclipse y lenguaje JAVA. Desde el entorno fue posible acceder a una funcionalidad de la herramienta que permite el reconocimiento de objetos sobre la interfaz de usuario web, a pesar que esta funcionalidad permite grabar el paso a paso de acciones sobre la interfaz de usuario; esto

[1] A pesar que Selenium es aplicable en todos los browser, su IDE, el cuál permite un fácil reconocimiento de los objetos sobre la interfaz, solo es integrable con Google Chrome y Mozilla.

no fue posible de realizar sobre el explorador Internet Explorer sobre el cual era necesario realizar la automatización de pruebas. A pesar que existen video tutoriales que explican el uso de la herramienta, estos son generados solo por la empresa que desarrolla LeanFT y no hay gran resolución de dudas en otros portales de la web.

- UIPath: Al descargar esta herramienta se cuenta con una interfaz sencilla de entender ya que tiene pocas funcionalidades. Para la realización de un caso de pruebas este se debe descomponer por componentes ejemplo: navegador web y sobre este componente definir la acción que se desea realizar, ejemplo: abrir navegador. A pesar que su aplicación resultó sencilla, la documentación para resolver dudas de sus funcionalidades como validación de resultados es muy poca, adicional a ello el caso de pruebas no fue posible de completar, ya que es obligatorio en el sistema allTRA la navegación entre ventanas, lo cual no fue posible de realizar con la herramienta.

En base a lo anterior se establece Selenium como la herramienta adecuada para la automatización de casos de prueba. Las pruebas de factibilidad tomaron un total de 40 horas/hombre.

5.2. Documentación de casos de prueba

Para la definición de los casos de pruebas se listaron cada una de las funcionalidades principalmente usadas por los clientes en el sistema, y se estableció un promedio de la cantidad de transacciones ejecutadas sobre esta funcionalidad por cliente (sobre los 3 últimos meses). Es importante detallar que no todas las funcionalidades se encuentran instaladas en todos los clientes ya que estas son configuradas a pedido. En la tabla a continuación se presenta el promedio de las transacciones por cliente.

Promedio Transacciones realizadas al mes						
Producto	Cliente 1	Cliente 2	Cliente 3	Cliente 4	Cliente 5	Cliente 6
Lettter of Credit for Import	525	298	345	100	321	90
Acceptance por Import	90	71	52	25	46	76

Deferred Payment for Import	37	71	96	28	15	17
Discount for Import	84	92	25	81	3	66
Letter of Credit for Export	18	39	7	31	73	39
Acceptance for Export	64	74	8	54	25	94
Deferred Payment for Export	73	13	35	44	62	40
Discount for Export	72	93	67	70	39	N/A
Letter of Credit Domestica	87	97	11	N/A	67	N/A
Acceptance for Domestica	99	1	77	N/A	98	N/A
Deferred Payment for Domestica	56	59	47	96	70	N/A
Discount for Domestica	34	49	16	93	28	N/A
Outgoing Stand By	6	73	40	98	94	N/A
Outgoing Stand By Doméstica	3	45	4	66	14	N/A
Incoming Stand By	18	98	11	28	78	N/A
Incoming Guarantees	82	22	85	43	77	44
Import Collections	36	40	85	80	95	81

Export Collections	84	11	89	71	99	82
Orden de Pago Enviada	66	91	1	58	82	82
Pago Saliente de Importación	N/A	18	97	N/A	96	N/A
Pago Saliente de Servicios	N/A	11	84	N/A	85	N/A
Outgoing Guarantees	95	71	25	44	19	20
Incoming Guarantees	96	89	81	75	33	82

Tabla 6. Promedio de cantidad de transacciones mensual por cliente

Gracias a esta información se establecieron los 3 clientes sobre los cuales se desarrollarían los casos automatizados, ya que cuentan con la mayor cantidad de productos y transacciones. La cantidad total de casos establecida es 479, distribuidos para cada cliente según la Tabla *7*, que se presenta a continuación.

	Cliente 1	Cliente 2	Cliente 3
EMISIONES	40	38	44
MODIFICACIONES	49	26	59
NEGOCIACIONES	39	43	42
UPDATE	32	34	33
TOTAL	160	141	178

Tabla 7. Casos de automatización
Fuente: Autor

Se generó un template para la documentación de los casos de prueba. Se estableció que la documentación de los casos de prueba debía ser en inglés con el propósito de recibir apoyo por parte de otros centros regionales.

Title:		Test Code: TCA-IM-ISS001
Summary:		
Pre-Conditions:		
Steps:		
No.	Steps:	Expected results:
1	Log in to the system with a correct user and password.	Screen: The workbasket should be displayed.

Tabla 8. Template para documentación de casos de pruebas.
Fuente: Autor

Se llevaron a cabo los siguientes pasos para el desarrollo de la documentación:

a. En reuniones el equipo de QA estableció el alcance de la automatización según los productos que tiene allTRA, no todos los casos se generaron ya que su repetición era baja.
b. Se conformó un equipo de automatización liderado por 1 analista de calidad con experiencia en automatización, e integrado por: analista de calidad con conocimiento funcional, arquitecto de software, ingeniero desarrollador.
c. Se comienza con la documentación de los casos de prueba haciendo uso del conocimiento previo sobre la estructura de la configuración del sistema. Se definen las diferentes bifurcaciones que puede tener un caso. Se establece la cantidad de casos que deberá tener los casos de prueba automatizados para el primer cliente.
d. La documentación es llevada a cabo por una analista de calidad en la herramienta Testlink.
e. Se identifica en los casos de prueba que los pasos de ejecución son repetitivos esto llevó a establecer la arquitectura del sistema de automatización con el objetivo de reutilizar los pasos.
f. Se establece como evidencia de ejecución de las pruebas capturas de pantalla en los pasos finales dentro de los casos de pruebas.

5.3.Definición de herramientas y arquitectura de clases

Previo al comienzo de la creación de los scripts de pruebas se definen las herramientas que integran el desarrollo y despliegue de las pruebas automatizadas, esto en base a las herramientas mayormente utilizadas por Surecomp en el desarrollo de software. A continuación, son listadas las herramientas:

- Maven: Utilizado para la compilación y despliegue de la aplicación, permite la configuración y versionamiento de las otras herramientas.

- Java: Como lenguaje de programación.

- Selenium: Driver de conexión encargado de realizar el control y las acciones sobre el navegador.

- JUnit: Utilizado para la realización de validaciones y compilador de evidencias mediantes logs.

- Internet Explorer: Único browser certificado para la aplicación allTRA.

Para la construcción de los casos se genera una arquitectura de clases la cual se muestra en el siguiente diagrama de paquetes (Figura 9). Los principales paquetes del sistema son: conexiones, donde se establecen las clases que permiten acceso a los motores de base de datos que se conectan con allTRA; test cases, sobre este paquete se alojan todas las clases son los script de pruebas, cada uno contiene el paso a paso y las respectivas validaciones; runner, este paquete contiene 2 grandes clases, la primera para la ejecución de las pruebas sobre el IDE y la segunda para la ejecución de pruebas por bloques en entorno de ejecución; el paquete util.client contiene las clases con los métodos que conocen las acciones que se deben realizar en cada paso de un caso de pruebas.

Para una mejor comprensión se explica los anterior mediante el siguiente ejemplo: para ejecutar un caso de pruebas se debe indicar su código según el nombre de la clase que se encuentra en el paquete test cases, esto en un archivo de texto plano (para bloques de casos estos deben ser separados por comas en dicho archivo). Luego al ser ejecutado del proyecto una clase en el paquete runner reconoce e instancia la clase, gracias a junit y la utilización de sus anotaciones el proyecto reconoce acciones que se realizan siempre sobre todos los casos como lo es el ingreso al sistema, esto se encuentra en el paquete util.client. Cada clase de caso de pruebas conoce su paso a paso cuya acción y validación también se encuentra en el paquete utils.cliente; esto permite la reutilización de pasos dentro de los diferentes casos. Como evidencia al finalizar cada uno de los pasos o cuando se encuentra una diferencia de validación el sistema realiza capturas de pantallas las cuales se almacenan en una carpeta con la fecha de la ejecución. También el sistema cuenta con un logger que permite revisar los resultados obtenidos divididos en 3 aspectos: warning con mensaje de descripción, succes mensajes de paso correcto y fail con los errores capturados por el sistema.

El sistema de pruebas cumple con los siguientes principios:

1. Cada caso de pruebas es 1 clase.
2. Se aplican las anotaciones de JUnit para definir los pasos: comunes previo al caso de pruebas, durante el caso y luego del caso de pruebas.
3. Cada caso está contenido en un paquete los cuales están divididos con la siguiente estructura TipoProducto.Evento.Cliente.
4. Cada cliente cuenta con un paquete utils que contiene las acciones que pueden ser realizadas en los casos de pruebas y una clase common que permite definir las acciones a realizarse al comienzo de un caso de pruebas.

5. Cada caso de prueba extiende de una clase con las funciones de los pasos y estas a su vez extienden de la función común por cliente.
6. Dentro de cada caso de pruebas se llama a la función la cual representa un paso y se envían las variables necesarias para la ejecución de la función.
7. Los casos a ser ejecutados son llamados por el nombre de cada clase, según una nomenclatura establecida. Estos son cargados mediante un archivo de texto plano.

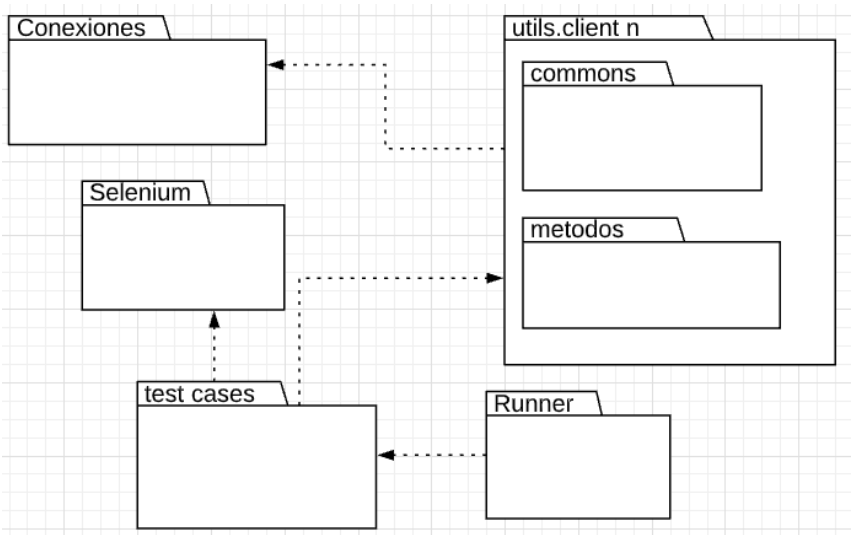


Figura 9. Diagrama de paquetes para las pruebas automatizadas de software
Fuente: Autor

5.4.Implementación de las pruebas

Una vez definida la arquitectura se dio paso a la construcción de los casos utilizando Eclipse como IDE. La metodología de implementación fue 1 cliente a la vez. Hasta no completar todos los casos del cliente no se daba paso a otro de ellos.

A continuación, se presenta un ejemplo de implementación de uno de los casos. El Listado de código 1 contiene parte de una de las clases del paquete test cases de este ejemplo es importante destacar las líneas 10 a 16 las cuales son variables utilizadas para instanciar componentes de logger que crean archivos planos de los resultados o variables con información de validaciones. La anotación @Test de la línea 17 hace parte de JUnit e indica el comienzo de los pasos del caso de pruebas. Desde la línea 19 se maneja cada paso comenzando por un comentario de código que describe el paso, seguido ejemplo línea 20 por el método que realiza este paso, para cada método se ha establecido como variable de estaba el nombre de la clase, el número de paso que ejecuta, y el driver que maneja la ejecución.

```
1 package com.surecomp.im.issuance.improsa;
2 import org.apache.log4j.Logger;
3 import org.junit.FixMethodOrder;
```

```
4 import org.junit.Test;
5 import org.junit.runners.MethodSorters;
6 import com.surecomp.utils.improsa.AlltraStepsMethodIMISS;

7 @FixMethodOrder(MethodSorters.NAME_ASCENDING )
8 public class TCAIMISS001 extends AlltraStepsMethodIMISS {
9 //AlltraStepsMethod asm = new AlltraStepsMethod();
10          String className =  this.getClass().getSimpleName();
11          private final Logger LOG = Logger.getLogger(className);
12          static final Logger successLog = Logger.getLogger("successLogger");
13          static final Logger errorLog = Logger.getLogger("errorLogger");
14          private String[] liabCodeValues1 = {"CCI LINEA","CCI VISTA","CCI
CONFIRM","LCI IMPORT","CONTINGENCIA","null"};
15          private static String tolerance="0";
16          private static String amountPrct="50";
17          private static String referenceID="TCAIMISS001"; // STEP-1.

18 @Test
19 public void testTCAIMISS001() {
20          //Select the new Import L/C option
21          selectNewImportLCOption(className,"2",driver);
22          //Select the options and get into the new Import L/C, the product type
should be IMPORT LETTER OF CREDIT (Carta crédito importación)
23          passSearchYFrame(className,"3",driver);
24          //Fill in the mandatory fields (amount, currency, expiry date, expiry
place, applicant, beneficiary, first advising bank, available with).
25          //The field available by should be NEGOTIATION
26          goToNewImportScreen(className,"4",driver);
27          getTransactionID(className,"4",driver);
28          validateStatusCheck(className,"4",driver);
```

Listado de código 1. Paso de un caso de pruebas

El Listado de código 2 muestra como ejemplo el código de una clase del paquete utils, donde se encuentran los métodos que ejecutan y validan los casos de pruebas, en su gran mayoría implementando selenium.

En concordancia con el código del listado 1, se presenta el método validateStatusCheck. Cada método contiene un inicio que se evidencia en el logger (ver líneas 2 y 3), luego mediante la instrucción try/catch se ejecuta el paso en este ejemplo en la línea 7 el driver de selenium busca y crea un objeto de la interfaz de usuario por el nombre, seguido por la extracción del texto de

este objetivo en la línea 8 y se valida si la extracción del texto genera una variable vacía de ser así el catch genera un logger de error en el caso contrario se da por concluido satisfactoriamente el paso.

```
1 public void validateStatusCheck(String className, String stepNum,
WebDriver driver) {
2	LOG.warn(className+" STEP No "+stepNum);
3	successLog.warn(className+" STEP No "+stepNum);
4	try {
5		LOG.warn("...validateStatusCheck");
6		// validateStatusCheck
7		Select statusSelect = new
Select(driver.findElement(By.name("boValue(im_orig status)")));
8		String statusValue =
statusSelect.getFirstSelectedOption().getText();
9		assertNotNull(statusValue);
10		LOG.warn("Status value : " + statusValue);
11		assertTrue(true);
12		LOG.warn(className+" SUCCESS | STEP No "+stepNum);
13		successLog.warn(className+" SUCCESS | STEP No
"+stepNum);
14		}
15	catch (Exception e) {
16		LOG.warn(className+" ERROR | STEP No "+stepNum);
17		errorLog.warn(className+" ERROR | STEP No "+stepNum);
18		assertTrue(false);
19		}
20 }
```

Listado de código 2. Ejemplo método que ejecuta un paso sobre un caso de pruebas

Otra implementación de código importante se realiza en el método común takeScreenshot, presentado parte de su código como ejemplo en Listado de código 3, para ello el análisis de los casos de pruebas permitió identificar que las evidencias de la ejecución se debían capturas siempre en las mismas pantallas del sistema por lo que el código ha sido reusable para todos los casos.

```
 1 public void takeScreenshot(String className, String stepNum, WebDriver
driver) {
2	LOG.warn(className+" STEP No "+stepNum);
3	successLog.warn(className+" STEP No "+stepNum);
```

```
4           try {
5                       LOG.warn("...takeScreenshot");
6                       //takeScreenshot
7                       LOG.warn("Saving screenshot of Accounting Entries");
8                       WebElement accEntries = driver.findElement
                                (By.xpath("//input[@title='Accounting Entries']"));
9                       accEntries.click();
10                      Thread.sleep(1000); //////////////////////TIME 3000
11                      File accEntriesScreen = ((TakesScreenshot)
driver).getScreenshotAs(OutputType.FILE);
12
13                      FileUtils.copyFile(accEntriesScreen, new
File(ConstantsMON.SCREENSHOT_URL1+ipFolder
                                +ConstantsMON.SCREENSHOT_URL2+
"lc\\"+this.getClass().getSimpleName()+"_accounting_entries.png"));
14                      Thread.sleep(1000);
```

Listado de código 3. método de capturas de evidencia median captura de pantallas

Durante el avance sobre la implementación de los casos se encontraron los siguientes retos o contingencias que se debieron resolver para completar la totalidad de los casos:

1. Para la ejecución de algunos casos se necesitaba información obtenida en casos anteriores. Esto se resolvió guardando el resultado del caso sobre archivos planos (ejemplo: código de una transacción que debe ser continuada en un segundo caso). Luego en el segundo caso se relaciona como variable el código de la clase del primero y así al iniciar la ejecución del segundo caso primero se consulta la información del documento. Se puede ver esta implementación en la línea 17 del Listado de código 1.
2. Los objetos de la interfaz de usuario no cuentan con id único, por lo que su identificación con Selenium era de un grado alto de complejidad. Adicional a ello, los nombres de los objetos varían con respecto de un cliente a otro ya que cada objeto lleva una abreviatura por cliente. Para esto se implementaron métodos de javascript que permiten ingresar a un elemento específico.
3. Se hizo necesaria la implementación de conexión a la base de datos del ambiente donde se ejecutaban los casos de pruebas, esto debido al uso de un solo usuario del sistema allTRA, al encontrar errores sobre el sistema el usuario permanecía con una sesión activa, lo cual no permitía continuar con la ejecución de próximos casos. Mediante la conexión a base de datos se estableció que al comienzo de cada caso se realizara un fin de sesión del usuario.
4. Para cada uno de los clientes fue necesario la generación de un script de configuración, esto con el fin de preparar información interna de cada ambiente de pruebas y la creación de un usuario en allTRA que contara con todos los permisos y accesos necesarios que requerían los casos de pruebas.

5. La aplicación allTRA hace uso de ventanas emergentes en donde algunas de ellas no podían ser identificadas por Selenium, para este caso se implementó la librería “Robot” de Java la cual permite según los pixeles de posición del objeto realizar acciones sobre la pantalla. La desventaja sobre esta implementación se hizo evidente tiempo después, ya que se identificó que algunos compilados posteriores cambiaban el posicionamiento de algunos objetos de la interfaz de usuario, por lo que fue necesario realizar cambios sobre casos de pruebas que ya había sido implementados.
6. Se implementó un archivo de tipo properties que permite cambiar información de configuración de una manera más ágil. Ejemplo: links de ambientes de pruebas sobre el cual debe operar las pruebas automatizadas.

5.5.Ejecución de las pruebas - evidencias

A medida que se contaba con la implementación de un grupo de casos estos eran ejecutados aleatoriamente por un analista de calidad con el objetivo de corroborar el cumplimineto de los pasos establecidos en los caos de prueba y la captura de evidencias, sin importr si el caso era o no existoso.

Para la ejecucón de un grupo de pruebas, se creó una clase donde mediante un archivo de configuración se identifica el cliente y el ambiente sobre el cual se deben ejecutar los casos; gracias a la nomenclatura definida para la identificación de los scripts de prueba; estos deben ser listados en un archivo de texto plano, serparados por comas (,). La clase anteriormente nombrada contiene estruturas de control (switch case) para cada cliente y con las primeras 5 letras del caso se reconoce la clase que contiene cada caso de pruebas.

Para la ejecución de los test automatizados no se requiró el uso de algún servidor de integración; esta se realizada mediante un .bat el cual se ejecutaba a demanda en general después de cada nuevo build que era instalado por parte del área de desarrollo. En la empresa no existe una cultura explicita DevOps.

Para la instalación del sistema se crea un manual de usuario, el cual fuera de fácil entendimiento para los analistas de calidad que no contaban con conocimientos técnicos.

Para la generación de resultados y evidencias de la ejecución de los casos se estableció un repositorio de carpeta donde se almacena capturas de pantalla y logs (warning, error y success). La distribución de estas carpetas es la siguiente: una carpeta por cliente, una carpeta por ambiente de pruebas, y una carpeta por cada fecha de ejecución ver Figura 10.

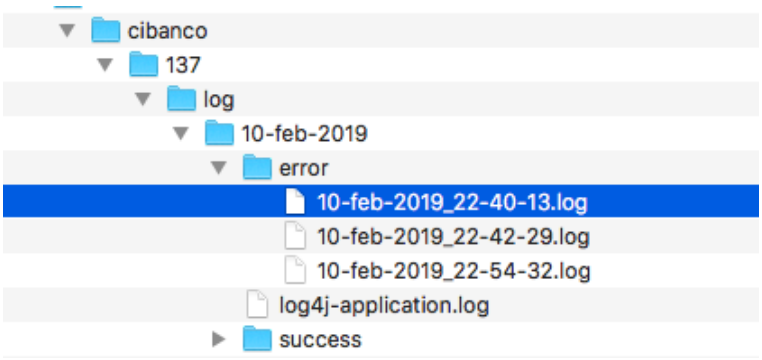


Figura 10. Organización de carpetas de evidencias
Fuente: Autor

El Listado de código 4 presenta un ejemplo de logs, el cual contiene la siguiente estructura: fecha y hora, tipo de log, clase de ejemplo y datos.

```
<2019-02-10 22:40:13> WARN  : AlltraCommonCIB cibanco.AlltraCommonCIB : --
------------------------------
<2019-02-10 22:40:13> WARN  : AlltraCommonCIB cibanco.AlltraCommonCIB : --
----------LOGS DATE-------------
<2019-02-10 22:40:13> WARN  : AlltraCommonCIB cibanco.AlltraCommonCIB :
10-feb-2019
<2019-02-10 22:40:13> WARN  : AlltraCommonCIB cibanco.AlltraCommonCIB : --
------------------------------
<2019-02-10 22:40:13> WARN  : AlltraCommonCIB cibanco.AlltraCommonCIB :
Properties file loaded correct
<2019-02-10 22:40:13> WARN  : AlltraCommonCIB cibanco.AlltraCommonCIB :
System author : Daniel Perez feat Maximiliano Mussuto feat Angela Cortes -
Surecomp 2018
<2019-02-10 22:40:13> WARN  : AlltraCommonCIB cibanco.AlltraCommonCIB :
Iniciando ejecucion
<2019-02-10 22:40:19> WARN  : AlltraCommonCIB cibanco.AlltraCommonCIB :
<2019-02-10 22:40:19> WARN  : AlltraCommonCIB cibanco.AlltraCommonCIB :
Resolución configurada en el monitor utilizado:
<2019-02-10 22:40:19> WARN  : AlltraCommonCIB cibanco.AlltraCommonCIB :
WIDTH = 1366   -   HEIGHT = 768
<2019-02-10 22:40:19> WARN  : AlltraCommonCIB cibanco.AlltraCommonCIB :
<2019-02-10 22:40:19> WARN  : AlltraCommonCIB cibanco.AlltraCommonCIB :
LA RESOLUCIÓN ES COMPATIBLE - SE INICIA APLICACIÓN
<2019-02-10 22:40:19> WARN  : AlltraCommonCIB cibanco.AlltraCommonCIB :
```

```
<2019-02-10 22:40:21> WARN  : AlltraCommonCIB cibanco.AlltraCommonCIB :
Server Selenium Ejecutandose
<2019-02-10 22:40:21> WARN  : AlltraCommonCIB cibanco.AlltraCommonCIB :
Landing screen : 387d184b-09af-4117-bcca-72700524f7a8
<2019-02-10 22:40:22> WARN  : AlltraCommonCIB cibanco.AlltraCommonCIB :
387d184b-09af-4117-bcca-72700524f7a8
<2019-02-10 22:40:22> WARN  : AlltraCommonCIB cibanco.AlltraCommonCIB :
39bb7abc-f073-4b55-983f-95f316a49035
<2019-02-10 22:40:22> WARN  : AlltraCommonCIB cibanco.AlltraCommonCIB :
Switching to : 39bb7abc-f073-4b55-983f-95f316a49035
<2019-02-10 22:40:22> WARN  : AlltraCommonCIB cibanco.AlltraCommonCIB :
TEST Loggin into alltra
<2019-02-10 22:40:22> WARN  : AlltraCommonCIB cibanco.AlltraCommonCIB :
Username : TESTUSER
<2019-02-10 22:40:22> WARN  : AlltraCommonCIB cibanco.AlltraCommonCIB :
Password : SURECOMP123
```

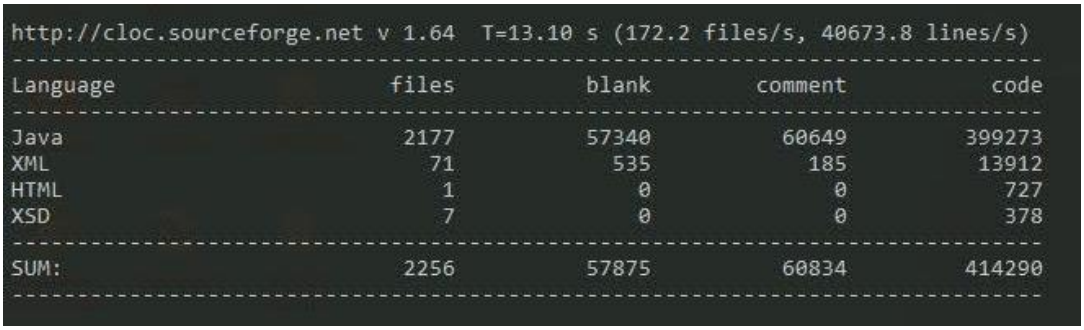
Listado de código 4. Resultado de logger de un caso de pruebas

Luego de la ejecución de un set de pruebas, un analista de calidad debe ingresar a la carpeta que corresponde a la fecha y hora de ejecución y revisar el archivo logger de tipo “fail”, el cual indica cuales casos han tenido fallas de validación. Con esta información podrá hacer uso de las capturas de pantalla para entender el comportamiento del fallo y así generar un ticket sobre el incidente o reportar el resultado a quienes estime conveiente.

Capítulo 6: Resultados sobre la implementación

En el siguiente capítulo se consolidan los resultados encontrados en la implementación del proyecto, mediante la captura de indicadores para medir el nivel de cumplimiento sobre los objetivos planteados; así como los resultados sobre una encuesta para conocer la relevancia del proyecto sobre el equipo de trabajo y sus impresiones.

La cobertura obtenida por las pruebas automatizadas es del 3,75% abarcando promedio de 15.000 de las aproximadamente 400.000 líneas de código JAVA del sistema allTRA, ver Figura 11. En términos funcionales se realizaron pruebas en 11 de las 23 principales funcionalidades del sistema. Es importante resaltar que la mayor parte de líneas de código del sistema son implementaciones de integración y comunicación con otros sistemas, no fue posible encontrar el porcentaje de líneas que corresponden a la base front del sistema que abarcan las pruebas de regresión.



```
http://cloc.sourceforge.net v 1.64  T=13.10 s (172.2 files/s, 40673.8 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Java                          2177          57340          60649         399273
XML                             71            535            185          13912
HTML                             1              0              0            727
XSD                              7              0              0            378
-------------------------------------------------------------------------------
SUM:                          2256          57875          60834         414290
-------------------------------------------------------------------------------
```

Figura 11. Cantidad de líneas del código base

La Tabla 9, contiene la cantidad de días de trabajo utilizados para la implementación de los casos de automatización.

Horas-hombre consumidos	Enero	Febrero	Marzo	Abril	Mayo	Junio	Total general
Cliente 1	128,26	90,1	128,35				346,71
Cliente 2				160,31			160,31
Cliente 3					159,8	7,99	167,79

Tabla 9. Horas-Hombre de implementación de casos
Fuente: Autor

	Cliente 1	Cliente 2	Cliente 3
EMISIONES	40	38	44
MODIFICACIONES	49	26	59
NEGOCIACIONES	39	43	42
UPDATE	32	34	33
TOTAL	160	141	178

Tabla 10. Cantidad de casos de implementación
Fuente: Autor

Con esta información presentada en la Tabla 10 sobre la cantidad total de casos implementados siendo en total 497 y con la presentada en la Tabla *7*, sobre la cantidad de casos por cliente se establece lo siguiente:

- Para el cliente 1, la implementación de un caso tomó en promedio un total de 2,16 horas-hombre.
- Para los clientes 2 y 3, la implementación de un caso tomó un promedio de 1,035 horas-hombre.
- La ejecución de un caso de pruebas toma en promedio de 2,03 minutos.

En comparación con los tiempos estimados por la jefatura de proyectos sobre la versión 2.8 reportados en la sección 1.3, donde se estimaba un promedio de 867,9 horas-hombre para la ejecución de pruebas de calidad, donde el 60% de este tiempo fue estimado para el consumo de pruebas de regresión. Con la implementación de pruebas automatizadas se redujo este porcentaje del 60% a 37,42% solo en tiempo de ejecución.

Sobre la herramienta se realizó una encuesta a los siguientes roles: Jefe del área de calidad, Desarrollador de software y Analista de calidad quienes estuvieron involucrados en el proceso de automatización de las pruebas. A cada una de ellas se les realizó las siguientes preguntas con el objetivo de conocer sus impresiones sobre los casos automatizados: ¿Qué impacto cree usted que tuvo la automatización sobre la organización?, ¿Se cumplieron las expectativas en el desarrollo de los casos?, ¿Qué le modificaría a la automatización?

A continuación, se detallan las respuestas entregadas por los encuestados a cada una de las preguntas:

Pregunta #1 - ¿Qué impacto cree usted que tuvo la automatización sobre la organización?

Jefe de Calidad: “En primera instancia nos ayudó a identificar los tipos de proyectos que tenemos, la similitud entre ellos y realizar pruebas de regresión en menor tiempo.”

Desarrollador: “La automatización es una herramienta muy útil en cualquier organización que cuenta con múltiples clientes y por ende, debe realizar instalaciones periódicas en distintos ambientes. Por lo tanto, aquellas tareas como pruebas de regresión que antes eran realizadas una y otra vez por personas de QA hoy en día pueden ser llevadas a cabo por un software ejecutable, permitiendo así destinar los recursos de QA a tareas específicas, dando mayor eficiencia al mayor activo que hoy en día podemos tener: el tiempo.”

Analista de Calidad: “En el ámbito de QA, es una herramienta que ha permitido la optimización de los tiempos de pruebas de regresión.”

Pregunta #2 - ¿Se cumplieron las expectativas con la automatización de los casos?

Jefe de Calidad: “Creo que inicialmente las expectativas eran mucho mas altas que las que tenemos actualmente, una vez que conocimos las dificultades que puede llegar a tener este proceso, las

expectativas se fueron emparejando con la realidad y podemos decir que sí se cumple lo que hoy en día pretendemos, tener pruebas de regresión automatizadas, lo que nos ahorra tiempo."

Desarrollador: "En su totalidad. Las pruebas fueron llevadas a cabo tal como las realizaría un profesional de QA, siguiendo los pasos lógicos para abordar con éxito un caso de prueba, con evidencia suficiente para respaldar la prueba (logs, screenshots). Actualmente es un caso de éxito para nuestro principal producto de comercio exterior en la región de América del Sur y Norte América, y se espera expandir a productos de otras regiones del mundo como EUROPA, dado su éxito en las regiones señaladas previamente."

Analista de Calidad: "Completamente. Los casos fueron adaptados a cada proyecto, resolviendo las diferentes dificultades presentadas de manera independiente."
Pregunta #3 - ¿Qué le modificaría a la automatización?

Jefe de Calidad: "En relación a lo que tenemos ahora, creo que la falencia está en el tipo de equipo que se utiliza, la versión del browser, por cuanto esa inestabilidad es la que le cambiaría."
Desarrollador: "Como todo proyecto de software inicial, siempre podrán existir mejoras. Algunas mejoras que personalmente consideraría serían: a) Independizar cada página de un sistema dentro de una clase (ej: LoginPage, LCIssuancePage, ScheduleMessagePage, etc), con tal de reutilizar cada uno de estos componentes en N clases de tipo test-cases abordadas mediante un framework de pruebas (JUnit, Test NG, u otro). b) Reemplazar Thread.sleep() por espera implícita (asignando un timeout para que cargue completamente el DOM) o explícita (esperar hasta detectar la presencia de un elemento sin cargar todo el DOM, también dando un tiempo límite de espera) mediante el WebDriver de Selenium. Las ventajas respecto a Thread.sleep() es que en las esperas implícitas o explicitas, si la carga se realiza en menos tiempo, la ejecución sigue de inmediato, mientras que con Thread.sleep() esperará la cantidad de milisegundos que le pasemos aunque haya cargado muy anticipadamente los elementos que buscamos.

Por lo general todo está muy bien, bastante parametrizable, haciendo bueno uso de archivos .properties para aspectos de configuración, y rutas lógicas en vez de rutas físicas."

Analista de Calidad: "Un ajuste en los tiempos de llenado de campos y búsqueda de transacciones para la liberación."

Como análisis de resultados sobre la encuesta se determina que a pesar que la herramienta de automatización fue bien recibida por cada uno de los integrantes del proyecto cada uno tiene una visión diferente sobre el impacto. Para los roles de desarrollador y analista un impacto positivo con expectativas cumplidas y para el jefe de área con una visión más objetiva sobre el proceso que implica automatización de casos de pruebas, quien reconoce que el esfuerzo puede ser mayor que el resultado esperado.

En relación a cada uno de los objetivos establecidos como resultado se obtiene lo siguiente: con respecto al objetivo específico #1: se definió e implementó un proceso formal de calidad de software para la empresa en donde se hizo partícipes a las demás áreas denotando que la calidad de software se encuentra intrínseca en todo el ciclo de vida de desarrollo. Como se muestra en la sección 4.3 se genera una ruta de procesos, detallados con objetivo, entradas, salidas y actividades claras para el trabajo diario del área.

Sobre el objetivo específico #2 se realizó la implementación de un sistema de automatización de pruebas ad-hoc a las necesidades de la empresa que permitió la liberación a tiempo de la versión seleccionada para las pruebas del presente proyecto.

El cumplimiento del objetivo específico #3, sobre el cual se debía reducir a un 30% el tiempo de ejecución de pruebas de software se vio afectado, en gran parte por el tiempo de respuesta de la aplicación allTRA, el cual debe generar cálculos matemáticos que toman tiempo y por ello se debió implementar algunos tiempos de espera sobre los casos de pruebas.

Capítulo 7: Conclusiones

En este capítulo se consolidan las conclusiones sobre el desarrollo del presente proyecto, a través de la formalización de un proceso para el área de calidad que se adapta a los objetivos y alcances definidos por el área; con apoyo de pruebas de regresión de aseguramiento de calidad automatizadas gracias a la construcción de un sistema que permite la implementación de las pruebas. Como principales entregables se obtiene el proceso formal del área y un total de 497 casos de regresión implementados en el sistema de automatización.

Descripción final

La empresa sobre la cual fue llevada a cabo la implementación, realiza la creación de productos informáticos para el intercambio de información financiera, siendo bancos el core principal de sus clientes. En ella se identificaron como principales problemas: informalidad sobre su proceso de calidad, el cual ha sido llevado manualmente y en base a la experiencia de los miembros que componen el equipo; lo que conllevaba a un alto uso de horas-hombre sobre tareas de ejecución de pruebas.

Para dar solución a las problemáticas se propuso definir mediante una metodología cuantitativa-cualitativa, la formalización e implementación un proceso de calidad de software, para ello se inició con la identificación del proceso que era llevado a cabo por el área, del cual se detallaron sus tareas, interacciones y roles que eran ejecutada de manera informal; gracias a esta actividad se conocieron los principales puntos de intervención dentro del proceso. Como segunda actividad, se realizó una evaluación de madurez del proceso mediante la ejecución de un cuestionario donde se evaluaron 19 subprocesos de calidad de software; siendo los subprocesos de herramientas de prueba, estrategias de pruebas y técnicas de especificación de pruebas, aquellos con el menor nivel de madurez realizados por el área.

Sobre esta base se desarrolla un primer modelo de proceso donde se agregan nuevas actividades como el análisis de documentación generada por otras áreas para determinar el alcance sobre las pruebas a realizar en un producto, la recepción y administración de ambientes de pruebas, la ejecución automatizada de pruebas de regresión. Este primer modelo fue evaluado de forma cualitativa mediante una reunión de retrospectiva de la cual se obtuvieron las siguientes ideas de mejora: División del proceso en sub-procesos que se deben ejecutar según el avance del proyecto, definición de un sub-proceso especial para la automatización de pruebas (en este punto se incorporaron 2 desarrolladores de software enfocados en la programación de las pruebas), modelamiento de governance para el área, definición de roles, entradas / salidas y acciones de cada sub-proceso.

A continuación se desarrolló una segunda versión del proceso, basado en la guía de mejores prácticas presentadas en el framework COBIT, cuya estructura se divide en dominios y procesos. Siendo los dominios agrupaciones de procesos que corresponden a una responsabilidad y los procesos son una serie de actividades controlables con objetivos claros y un resultado medible. Como mejora se planteó la división del proceso en 4 grandes dominios: (1) planificación y organizar, (2) adquirir e implementar, (3) entregar y dar soporte (4) monitorear y evaluar.

Para dar respuesta a la extesión sobre plazos de entrega, como soporte al subproceso de adquirir e implementar se realizó la implementación del sistema de automatización de pruebas de software, para el tipo de pruebas de regresión; para la determinación de una herramienta se llevaron a cabo pruebas de concepto para el desarrollo de los casos de pruebas, siendo Selenium la herramienta seleccionada al ser de código abierto, aplicación en todos los browser y conocimiento previo por parte del equipo. Luego se definió una arquitectura de despliegue del sistema y de clases la cual permite el reuso de componentes para una implementación más ágil de los casos.

En conjunto el área de calidad establece el alcance de las pruebas, los clientes sobre los cuales se implementaran los casos, la conformación de un equipo de automatización liderado por un analista de calidad con experiencia en automatización (autora del presente documento), e integrado por: analista de calidad con conocimiento funcional, arquitecto de software, ingeniero desarrollador. Para la documentación de los casos se define un template documento formalizado en la versión definitiva del proceso para el área; donde se detalla paso a paso de un caso de pruebas y cada uno de los resultados esperados dando paso a la construcción de los scripts de pruebas.

Resultados y cumplimiento de objetivos

Como se muestra en la sección de resultados, la curvatura de aprendizaje sobre la automatización de pruebas fue más rápida a medida que avanzó el tiempo, se logró disminuir el tiempo de desarrollo de una prueba casi a la mitad del tiempo. Sobre la ejecución de los casos automatizados se detectaron puntos de mejora a medida que se generaban nuevos builds del sistema sobre el que se ejecutan las pruebas, ya que algunos de ellos contenían cambios de interfaz de usuario donde algunos de los objetos visuales cambiaban de posición, haciendo que el caso automatizado no pudiera ejecutar alguna acción o validación sobre el objeto y fue necesario la actualización sobre estos casos. Al momento de ejecución de los casos también fue evidente la ralentización en el tiempo de ejecución de los casos esto se debía a un alto tiempo de respuesta por parte del servidor donde se encuentra la aplicación allTRA, por lo que fue necesario implementar tiempos de espera dentro de los mismos casos de prueba.

Sobre el cumplimento de los objetivos, se logró definir un proceso formal para el área de calidad e la empresa. Se logró la implementación del sistema de pruebas automatizadas para el apoyo de las tareas de ejecución de pruebas de regreso. Aunque no se logró la reducción de las horas-hombre de ejecución de pruebas de regresión a un 30% la reducción a un 37% fue considerable para la empresa; como se describe en la sección de resultados gracias a la encuesta realizada al equipo sobre el sistema de automatización.

Como se denota en la encuesta realizada se cumplieron con las expectativas inicialmente presentadas, entregando valor y mejorando la calidad de los procesos en el área de calidad.

Contar con un sistema de automatización le permitió a la empresa crear nuevos modelos de negocio, ya que se abrió el desarrollo de pruebas automatizadas como servicio a otras sedes de la compañía, actualmente se esta realizando automatizaciones para la sede de: Israel y Alemania.

Conclusiones personales

Si bien la idea de automatización de pruebas surge como una necesidad latente para la empresa, no era claro el alcance de la herramienta por parte de las jefaturas de la empresa; esto se hizo notorio

al momento de tomar decisiones sobre requerimientos del sistema de automatización como por ejemplo la forma de capturar evidencias; adicional a una alta expectativa de solución, sin considerar la inversión de tiempo que se debía realizar.

Como experiencia personal, el desarrollo del sistema me permitió generar nuevos conocimientos técnicos como la definición de arquitecturas, la implementación de nuevas herramientas y aumentar mis concocimientos sobre el desarrollo de software. Algunos de los conceptos utilizados como procesos, herramientas del área de calidad fueron términos que aprendí en el transcurso de las materias estudiadas dentro del Magíster por lo que estos fueron una base de conocimiento importante.

Tuve la oportunidad de liderar por primera vez un proyecto de implementación donde mis conocimientos e ideas eran tomadas en cuentas y pude ver de primera mano los cambios culturales que surgieron sobre la organización y sobre mi equipo de trabajo.

Recomendaciones a futuro

Como recomendaciones a futuro, es importante que la empresa prepare sus sistemas en relación a la generación de pruebas automatizadas; esto se puede lograr implementando códigos únicos sobre los objetos de la interfaz de usuario que permitan la fácil identificación y detección de los mismos por parte de las herramientas.

Para el sistema de automatización se recomienda la implementación de resultados que sean entendible por cualquiera de los roles que hacen parte del proceso de desarrollo de software, los que actualmente se detallan en un logger y capturas de pantalla; para ello se podría integrar librerías o framework de reportes. Adicional, se puede incluir el uso de un servidor de integración como por ejemplo Jenkis donde a medida que se registre un nuevo build del software se realice la ejecución automática de las pruebas.

Bibliografía

[1] S. H. Gérald Lomprey, "La importancia de la calidad en el desarrollo de productos de software," Facultad de Ingeniería y Tecnología, Universidad de Montemorelos, México , 2008.

[2] S. Gorris., "¿Qué es el código SWIFT o BIC de una cuenta bancaria?," 2018. [Online]. Available: http://cibergestion.com/que-es-el-codigo-swift-o-bic-de-una-cuenta-bancaria/.

[3] A. M. d. P. Internacional, "Actualización estandaes 2018," Noviembre 2018. [Online]. Available: http://www.mediosdepagointernacional.es/noticias/entradasintitulo-5.

[4] Universidad de los Andes, Métodos de desarrollo de Software, 2011.

[5] C. Alonso-Torres, "Orientaciones para implementar una gestión basada en procesos," Ingeniería Industrial, 2014.

[6] C. M. C. I. Pablo Andrés Vaca, "Test-Driven Development - Una aproximación para entender su," Universidad Tecnológica Nacional, Facultad Regional Córdoba, 2016.

[7] ISO, "ISO25000 - Calidad del producto," 2019. [Online]. Available: https://iso25000.com/index.php/normas-iso-25000/iso-25010.

[8] R. Pressman, Ingeniería del software, Mc Graw Hill, 1998.

[9] M. Callejas, A. Alarcón and A. Álvarez, "Modelos de calidad del software, un estado del arte," Ingeniería y Tecnología, p. 15, 2017.

[10] IT Governance Institute, Cobit 4.0, Rolling Meadows, 2007.

[11] J. A. Baldonedo, "Modelo CMMI y métodos ágiles en la gestión de proyectos software," Universidad de Oviedo, 2017.

[12] I. S. T. Q. Board, Certified Tester, 2018.

[13] N. Kumar, "How Will TestLink Optimize Your Testing Workflows?," [Online]. Available: https://www.tadigital.com/blog/testlink-optimize-testing-workflows/.

[14] G. S. Mares, "Selenium WebDriver en un Ambiente de Pruebas Continuas," [Online]. Available: https://sg.com.mx/revista/54/selenium-webdriver-un-ambiente-pruebas-continuas.

[15] About RPA, "UIPath," 2019. [Online]. Available: https://www.uipath.com/.

[16] J. A. D. Insha Altaf, "2015 International Conference on Green Computing and Internet of Things (ICGCIoT)," in 2015 International Conference on Green Computing and Internet of Things (ICGCIoT).

[17] R. T. Nicey Paul, "An Approach of Automated Testing on Web Based Platform Using Machine Learning and Selenium," in 2018 International Conference on Inventive Research in Computing Applications (ICIRCA), 2018.

[18] E. Serna, R. Martínez and P. A. Tamayo, "Un modelo para determinar la madurez de la automatización de las pruebas del software como área de investigación y desarrollo," IEEE, 2017.

[19] T. Troncos, "Sistema de validaciòn para el desarrollo incremental de un intérprete de R en COQ," 2018.

[20] Surecomp, "Aquitectura técnica allTRA," Surecomp, 2019.

[21] P. O, "Modelo de Verificación y Validación Basado en CMMI," Investigación E Innovación En Ingenierías, 2013.

[22] A. G. Ridley Tasmania Univ., J. Young and P. Carroll, "IEEE," [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1265566.

[23] E. Dustin, "The Automated Testing Life-cycle Methodology (ATLM)," 2000.

[24] Surecomp, "Acerca de," 2018. [Online]. Available: https://www.surecomp.com/es/acerca-de-nosotros/mision-vision-y-valores/.

[25] Fundación Cardiovascular de Colombia , "Indicador actividades planeados vs. actividades estimadas," Floridablanca, 2015.

[26] Ágril Ibérica, "¿Qué hacemos con recursos humano en Agile?," 17 Febrero 2013. [Online]. Available: http://agilib.org/2013/02/17/que-hacemos-con-recursos-humanos-en-agile/. [Accessed 10 Octubre 2015].

[27] O. d. p. d. informática, "PMOinformatica," [Online]. Available: http://www.pmoinformatica.com/2015/04/herramientas-gestion-calidad-software.html.

[28] D. Carrizo and A. Alfaro, "Método de aseguramiento de la calidad en una metodología de desarrollo de software: un enfoque práctico," Ingeniare. Revista chilena de ingeniería, 2018.

Anexo A – Horas del proyecto de actualización de versión 2.6 para Itau Paraguay

Esta tabla corresponde al detalle de horas totales que constituyó el proyecto versión 2.6 para el cliente Itau Paraguay, del cual se realizó la estimación de tiempos de pruebas para la versión 2.8. Se destacan en amarillo las actividades correspondientes a calidad de software.

Row Labels	Suma de Actual in Days	Suma de No of hours
SULA ITAU PY ALLTRA UPGRA	2511	22477
Account Management	19,26	174,75
allNETT Implementation	31,53	284
allNETT SSO Interface	2	18
allNETT Support	6,58	59
allTRA DWH Interface	1,98	18
allTRA Func Detailed Spec	156,78	1409
allTRA FX Rates Interface	1,98	18
allTRA Implementation	311,42	2799
Documentation	8,54	77
Environment Setup Maint	6,87	62
Management	13	118
Meetings Reviews	14,29	121
Packing/Installation	15,76	142,5
Programming	114,27	1031
QA Test Design	29,96	268,5
QA Test Execution	74,61	671
QA Test Report	13,32	121
(blank)	20,8	187
allTRA Interfaces Development	134,42	1171,25
Documentation	1,91	17,25
Meetings Reviews	13,46	118,75
Product Support	0,13	1
Programming	73,79	641
QA Test Design	0,03	0,25
QA Test Execution	12,52	100
Technical Specification	32,58	293
allTRA Support	1001,78	8968,25
Documentation	3,24	29
Environment Setup Maint	17,45	157,5
Infrastructure Programmin	0,66	6
Management	14,96	132,5
Meetings Reviews	47,36	401,25
Packing/Installation	10,57	95,5

Product Support	22,4	201,5
Programming	660,42	5926,75
QA Regression	2,39	21,25
QA Test Execution	44,05	392,5
Unit Testing	178,28	1604,5
CR for Accounting Enhancements	34,93	297,5
Documentation	2,68	24
Functional Specification	3,77	34
Meetings & Reviews	10,47	83,5
Programming	16,89	146
QA Test Execution	1,12	10
Gathering Information SOW	149,83	1348
Documentation	18,72	169
Estimations	2,27	20,5
Functional Specification	108,77	979
Meetings Reviews	2,52	22,5
Technical Specification	16,42	148
(blank)	1,13	9
Maintenance allTRA	50,48	456
Environment Setup & Maint	36,1	326
Product Support	13,93	126
QA Test Execution	0,45	4
PHASE 1 ON SITE SUPPORT	14,26	128,5
QA Test Execution	9,56	86
Unit Testing	4,7	42,5
Project Management	225,37	2032
Documentation	4,09	37
Management	172	1551
Meetings Reviews	49,28	444
Prueba FileServer Linux	4,83	43,5
Documentation	0,39	3,5
FILE SERVER TESTS	4,44	40
QA allTRA - allNETT	322,07	2900,25
Documentation	12,24	110,25
Meetings Reviews	1,17	10,5
QA Test Design	6	54
QA Test Execution	297,82	2682
QA Test Report	4,84	43,5
Technology Support	0,67	6
Environment Setup & Maint	0,67	6
WO LCI Collateral Prepago	14,78	127
Functional Specification	1,88	17

Meetings & Reviews	2,77	22
Packing/Installation	0,5	4
Programming	4,73	40
QA Test Execution	4,9	44
WO SAM PART II	26,05	219
Functional Specification	1,56	14
Management	7,68	61
Meetings & Reviews	0,56	5
Programming	7,27	58
QA Test Design	4,06	36,5
QA Test Execution	2,93	26,5
Unit Testing	1,99	18

Anexo B – Horas del proyecto de actualización de versión 2.6 para Monex

Esta tabla corresponde al detalle de horas totales que constituyó el proyecto versión 2.6 para el cliente Itau Monex, del cual se realizó la estimación de tiempos de pruebas para la versión 2.8. Se destacan en amarillo las actividades correspondientes a calidad de software.

SULA MONEX UPG ALTRA 2.6	748,99	6654,25
Technology Support	11,29	102
UPG allNETT Implementation	181,73	1636,75
Documentation	13,73	123,5
Environment Setup & Maint	2,11	19
Infrastructure Programmin	37,09	334
Management	0,12	1
Meetings & Reviews	4,69	42,5
Packing/Installation	0,33	3
Product Support	2,84	25,5
Programming	120,71	1087,25
QA Test Execution	0,11	1
UPG allNETT Ongoing Support	3,62	32,5
UPG allTRA Implementation	282,31	2452,25
UPG allTRA Onging Support	3,51	31,5
UPG Onsite Support	20,21	182,5
UPG Project Management	43,75	393,5
UPG QA allTRA - allNETT	202,57	1823,25
Documentation	26,25	237
Environment Setup & Maint	0,69	6,25
Management	2,85	25,5
Meetings & Reviews	8,26	73
QA Test Design	3,5	31,5
QA Test Execution	160,63	1446,5
QA Test Plan	0,39	3,5

Anexo C – Cuestionario medición madurez del proceso

En este anexo se presenta la puntuación registrada para la medición de madurez del proceso presentada en la sección 4.1.

#	Description	#NA	#?	Score	P1
1	Test strategy				
1.A	Strategy for single high-level test (A)				
1.A.1	A motivated consideration of the product risks takes place, for which knowledge of the system, its use and its operational management is required			8	8
1.A.2	There is a differentiation in test depth, depending on the risks and, if present, the acceptance criteria: not all subsystems are tested equally thoroughly and not every quality characteristic is tested (equally thoroughly)			6	6
1.A.3	One or more test specification techniques are used, suited to the required depth of a test			3	3
1.A.4	For retests also, a (simple) strategy determination takes place, in which a motivated choice of variations between 'test solutions only' and 'full retest' is made			3	3
1.B	Combined strategy for high-level tests (B)				
1.B.1	Coordination takes place between the different high-level tests, often the system, acceptance, and production acceptance tests, in the field of test strategy (risks, quality characteristics, area of consideration of the test, and planning)			3	3
1.B.2	The result of the coordination is a coordinated strategy, which is put in writing. During the total test process, this strategy is controlled.			3	3
1.B.3	Each high-level test determines its own test strategy, based on the coordination strategy, as described in level A.			3	3
1.B.4	Deviations from the coordinating strategy are reported, after which a substantiated adjustment to the coordinating strategy is made, based on the risks.			2	2
1.C	Combined strategy for high-level tests plus low-level tests or evaluation (C)				

1.C.1	Coordination takes place between the high-level tests and the low-level tests or the evaluation levels in the area of test strategy (risks, quality characteristics, area of consideration of the test/evaluation, and planning).			3	3
1.C.2	The result of the coordination is a coordinated strategy, which is documented. During the total (evaluation and) test process this strategy is controlled.			3	3
1.C.3	Each high-level test determines, on the basis of the coordination, its own test strategy, as described in level A.			3	3
1.C.4	(if applicable) Each low-level test determines, on the basis of the coordination, its own test strategy, as described in key area Low-level testing, level C.	1			NA
1.C.5	(if applicable) Each evaluation level determines, on the basis of the coordination, its own evaluation strategy, as described in key area Evaluation, level B.	1			NA
1.C.6	Deviations from the coordinating strategy are reported, after which, on the basis of the risks, a substantiated adjustment of the coordinating strategy is made.			4	4
1.D	Combined strategy for all test and evaluation levels (D)				
1.D.1	Coordination takes place between the high-level tests, the low-level tests, and the evaluation levels in the area of test strategy (risks, quality characteristics, area of consideration of the test/ evaluation, and planning).			3	3
1.D.2	The result of the coordination is a coordinating strategy, which is documented. During the total evaluation and test process this strategy is controlled.			3	3
1.D.3	Each high-level test determines its own test strategy on the basis of the coordination, such as described for level A.			3	3
1.D.4	Each low-level test determines its own test strategy on the basis of the coordination, such as described in the key area Low-level testing, level C.	1			NA
1.D.5	Each evaluation level determines its own evaluation strategy on the basis of the coordination, such as described in the key area Evaluation, level B.	1			NA

1.D.6	Deviations from the coordinating strategy are reported, after which a substantiated adjustment of the coordinating strategy is made on the basis of the risks.			4	4
2	Life-cycle model				
2.A	Planning, Specification, Execution (A)				
2.A.1	For the test (at least) the following phases are distinguished: planning, specification, and execution. These are subsequently performed, possibly per subsystem. A certain overlap between the phases is allowed.			5	5
2.A.2	Activities to be performed for the planning phase are: formulate assignment, determine the test basis, determine test strategy, set up organization, set up test deliverables, define infrastructure and tools, set up management, determine planning, produce test plan			5	5
2.A.3	Activities to be performed for the specification phase are: design test cases and test scripts, specify intake of test object and infrastructure, realize test infrastructure			4	4
2.A.4	Activities to be performed for the execution phase are: take in test object and infrastructure, set up starting test databases, execute (re)tests			4	4
2.B	Planning, Preparation, Specification, Execution, and Completion (B)				
2.B.1	For high-level tests the following phases are distinguished: Planning, Preparation, Specification, Execution, and Completion. The phases are executed consecutively, possibly per subsystem. A certain overlap between the phases is allowed.			5	5
2.B.2	Activities to be performed during the preparation phase are: inspection of test base			5	5
2.B.3	Activities to be performed during the completion phase are: evaluate test object, evaluate test process, formulate end report, conserve testware			5	5
3	Moment of involvement				
3.A	Completion of test basis (A)				
3.A.1	The activity 'testing' starts simultaneously with or earlier than the completion of the test basis for a restricted part of the system that is to be tested separately			7	7
3.A.2	(The system can be divided into several parts which are finished, built, and tested separately. The testing of the first subsystem			7	7

	has to start at the same time as or earlier than the completion of the test bases of that particular subsystem)				
3.B	Start of test basis (B)				
3.B.1	The activity 'testing' starts simultaneously with or earlier than the phase in which the test basis (often the functional specifications) is defined			3	3
3.C	Start of requirements definition (C)				
3.C.1	The activity 'testing' starts simultaneously with or earlier than the phase in which the requirements are defined			7	7
3.D	Project initiation (D)				
3.D.1	Then project is initiated, the activity 'testing' is also started			3	3
4	Estimating and planning				
4.A	Substantiated estimating and planning (A)				
4.A.1	The test estimating and planning can be substantiated (so not just 'we did it this way in the last project')			6	6
4.A.2	In the test process, estimating and planning are monitored, and adjustments are made if needed			3	3
4.B	Statistically substantiated estimating and planning (B)				
4.B.1	Metrics concerning progress and quality are structurally maintained (on level B of the key area Metrics) for multiple, comparable projects			7	7
4.B.2	This data is used to substantiate test estimating and planning			7	7
5	Test specification techniques				
5.A	Informal techniques (A)				
5.A.1	The test cases are specified by means of a described technique			5	5
5.A.2	This technique requires at least the description of: (a) the starting situation, (b) the change process = test actions to be performed, and (c) the expected end result			5	5
5.B	Formal techniques (B)				
5.B.1	Besides informal techniques, formal techniques are also used, providing unambiguous ways of getting from the test basis to test cases			5	5
5.B.2	A substantiated judgment is possible about the level of coverage of the collection of test cases (compared to the test basis)			3	3

5.B.3	The testware is reusable (within the test team) by means of a uniform working method			3	3
6	Static test techniques				
6.A	Inspection of test basis (A)				
6.A.1	Preceding the definition of the test cases, a study of the testability of the test basis is performed			6	6
6.A.2	In this study checklists are used			6	6
6.B	Checklists (B)				
6.B.1	Static tests other than the inspection of the test basis take place by means of checklists (approved by project and/or customer)			6	6
7	Metrics				
7.A	Project metrics (product) (A)				
7.A.1	In the (test) project Input metrics are recorded for used resources (hours)			8	8
7.A.2	In the (test) project Input metrics are recorded for performed activities (effort and lead time)			8	8
7.A.3	In the (test) project Input metrics are recorded for size and complexity of the tested system (in function points, number of functions, and/or building effort)			8	8
7.A.4	In the (test) project Output metrics are recorded for test products (specifications and test cases, log reports)			8	8
7.A.5	In the (test) project Output metrics are recorded for test progress (performed tests, status (finished/not finished))			8	8
7.A.6	In the (test) project Output metrics are recorded for number of defects (defects by test level, by subsystem, by cause, priority, status (new, in solution, corrected, retested))			8	8
7.A.7	The metrics are used in test reporting			8	8
7.B	Project metrics (process) (B)				
7.B.1	In the (test) project Result measurements are made for at least two of the following items: ● defect find-effectiveness (defects found versus total defects present; analyze which previous test should have found the defects) ● defect find-efficiency (defects found per hour spent) ● test coverage level (test targets covered by a test case compared to the number of possible test targets) ● testware quality ('defects' found whose cause turned out to be wrong testing,			6	6

	compared to total number of defects found ●perception of quality (by means of reviews and interviews of users, testers, and other people involved)				
7.B.2	Metrics are used in test reporting			8	8
7.C	System metrics (C)				
7.C.1	The metrics mentioned above are recorded for development, maintenance, and production			8	8
7.C.2	Metrics are used in the assessment of the effectiveness and efficiency of the test process			6	6
7.D	Organization metrics (>1 system) (D)				
7.D.1	Organization-wide mutually comparable metrics are maintained for the already mentioned data			8	8
7.D.2	Metrics are used in assessing the effectiveness and efficiency of the separate test processes, to achieve an optimization of the generic test methodology and future test processes			6	6
8	Test tools				
8.A	Planning and control tools (A)				
8.A.1	Automated tools (other than standard word processing) are used for the defect administration and for at least two other activities of Planning and Control			1	1
8.B	Execution and analysis tools (B)				
8.B.1	At least two sorts of automated tools are used for test execution, such as Capture & Playback tools, test coverage tools, etc.			1	1
8.B.2	The test team has a general insight into the cost/profit ratio of these tools			1	1
8.C	Extensive automation of the test process (C)				
8.C.1	Automated tools (other than standard word processing) are used for the planning phase (for the activities estimating, planning, progress monitoring, configuration management, and defect administration), preparation, specification, and execution (in total at least five sorts of tool should be used)			1	1

8.C.2	The test team has a general insight into the cost/profit ratio of these tools				
9	Test environment				
9.A	Managed and controlled test environment (A)				
9.A.1	Only with the permission of the test manager can changes and/or deliveries take place in the test environment			8	8
9.A.2	The environment must be set up in time			8	8
9.A.3	The test environment is managed (with regard to setup, availability, maintenance, version management, error handling, authorizations, …)			8	8
9.A.4	The saving and restoring of certain test situations can be arranged quickly and easily			6	6
9.A.5	The environment is sufficiently representative for the test to be performed, which means: the closer the test level is to production, the more the environment is 'as-if-production'			8	8
9.B	Testing in the most suitable environment (B)				
9.B.1	Each test is performed in the most suitable environment, either by execution in another environment or by quickly and easily adapting the own environment			8	8
9.B.2	The environment is finished in time for the test and there is no disturbance by other activities during the test			7	7
9.B.3	The risks taken are analyzed and adequate measures are taken			6	6
9.C	Environment on call (C)				
9.C.1	The environment that is most suited for a test is very flexible and can quickly be adapted to changing requirements			6	6
10	Office environment				
10.A	Adequate and timely office environment (A)				
10.A.1	The office infrastructure needed for testing (offices, meeting rooms, telephones, PCs, network connections, office software, printers, …) is arranged on time			8	8
10.A.2	Things related to office organization have a minimal impact on the progress of the test process (as little moving as possible, physical distance between testers and the rest of the project not too large, …)			8	8
11	Commitment and motivation				
11.A	Assignment of budget and time (A)				

11.A.1	Testing is regarded by personnel involved as necessary and important			7	7
11.A.2	An amount of time and budget is allocated to testing			7	7
11.A.3	Management controls testing based on time and money. A feature is that if the test time or budget is exceeded, initially a solution is sought within the testing (doing overtime or employing extra people when exceeding time limits or on the contrary cutting time and/or budget).			7	7
11.A.4	In the team there is enough knowledge and experience in the testing.			5	5
11.A.5	The activities for testing are full-time for most participants (therefore there are not many conflicts with other activities).			7	7
11.A.6	There is a good relationship between the testers and other disciplines in the project and the organization.			7	7
11.B	Testing integrated in project organization (B)				
11.B.1	All those involved find that testing has a noticeable positive influence on the quality of the product.			7	7
11.B.2	The management wants to have insight into the depth and quality of testing.			6	6
11.B.3	The management controls testing based on time, money, and quality. A feature is that the solution to test problems (for example, exceeding test time or budget) is also sought outside the test project. Possibly the developer is addressed here.			6	6
11.B.4	In the project planning, the cycle of testing, rework, and retesting is taken into account.			7	7
11.B.5	Testing has a say in the delivery sequence of the developer.			4	4
11.B.6	The advice from testing is discussed in project meetings.			8	8
11.C	Test engineering				
11.C.1	The test team is involved in the design and realization to provide optimal testability of the system ('design for test').			2	2
11.C.2	The test team has sufficient knowledge and skills to provide a meaningful realization of the checkpoint mentioned above.			5	5
11.C.3	The recommendations of the test team are considered 'seriously' by the organization and/or project.			5	5

11.C.4	Management supports testers (with people and means) in working continually on the improvement of the test process.			7	7
11.C.5	Participation in testing is regarded as a 'promotion'; testing has a high status.			5	5
11.C.6	The development process is of sufficient maturity: at least time and quality are controlled.			5	5
11.C.7	Test jobs are described at an organization level, including career possibilities and reward structures.			6	6
12	Test functions and training				
12.A	Test manager and testers (A)				
12.A.1	The test personnel consist at the very least of a test manager and a number of testers.			7	7
12.A.2	The tasks and responsibilities have been defined.			7	7
12.A.3	The test personnel has had specific test training (for example, test management, test techniques, etc.) or has sufficient experience in the field of testing.			7	7
12.A.4	For the acceptance test, expertise in the subject matter is available to the test team.			8	8
12.B	(Formal) Methodical, Technical, and Functional support, Management (B) of test process, testware and infrastructure				
12.B.1	The task Methodical Support is outlined separately. Its activities are defining and maintaining test instructions, procedures, and techniques and advising about and evaluating the correct application of the above.			8	8
12.B.2	The task Technical Support is outlined separately.			8	8
12.B.3	The task Functional Support is outlined separately.			8	8
12.B.4	The task Management test process is outlined separately and is responsible for the registration, storage, and availability of all management objects of the test process. Sometimes one will carry out the management oneself, ill other cases one will organize and/or evaluate that management. Objects to be managed are progress, budgets, and defects.			8	8

12.B.5	The task Management testware is outlined separately and is responsible for the registration, storage, and availability of all management objects of the testware. Sometimes one will carry out the management oneself, in other cases one will organize and/or evaluate that management. Objects to be managed are test documentation, test basis, test objects (internal), test cases including test files and databases, test instructions and procedures.			7	7
12.B.6	The task Management test infrastructure is outlined separately and is responsible for the registration, storage, and availability of all management objects of the test infrastructure. Sometimes one will carry out the management oneself, in other cases one will organize and/or evaluate that management. Objects to be managed are test environments (test databases) and test tools.			8	8
12.B.7	The persons who carry out these tasks have sufficient knowledge and experience.			6	6
12.B.8	The time needed for these tasks is planned. Supervision is carried out to see that these tasks are in fact performed.			7	7
12.C	Formal internal Quality Assurance (C)				
12.C.1	Parallel to the test plan, an internal QA plan for testing is formulated			4	4
12.C.2	The person assigned the QA task has no other tasks within the test team			4	4
12.C.3	The results of the QA activities are used as input for further test process improvement			4	4
12.C.4	The person who performs the QA task has sufficient QA knowledge and experience			6	6
13	Scope of methodology				
13.A	Project specific (A)				
13.A.1	A methodology is formulated for each project			5	5
13.A.2	The aspects described cover at least: a description of the full life cycle model of testing, management of the test process (progress and quality), test product management, and test specification techniques to be used			7	7
13.A.3	The methodology is followed			7	7
13.B	Organization generic (B)				

13.B.1	The methodology is defined in a generic model for the organization			5	5
13.B.2	Each project works according to this generic model			5	5
13.B.3	Variances are sufficiently argued and documented			6	6
13.C	Organization optimizing, R&D activities (C)				
13.C.1	There is structured feedback process (both formally elicited and actioned by the R&D department) in the generic model			7	7
13.C.2	Structural maintenance and innovation (R&D) are done on the generic model, for example on the basis of feedback			8	8
14	Communication				
14.A	Internal communication (A)				
14.A.1	There is a periodic meeting within the test team. This meeting has a fixed agenda and its main focus is progress (lead time and effort spent) and the quality of the object to be tested			8	8
14.A.2	Periodically, each team member participates in the meeting			8	8
14.A.3	Deviations from the test plan are communicated and documented			6	6
14.B	Project communication (defects, change control) (B)				
14.B.1	In the test team meeting minutes are taken.			4	4
14.B.2	In the test team meeting, besides progress and the quality of the quality of the test object, the quality of the test process is a fixed subject on the agenda.			4	4
14.B.3	Periodically, the test manager reports progress and the quality of the object to be tested, including the risks, in the project meeting. The test manager also reports the quality of the test process.			8	8
14.B.4	Agreements in this meeting are documented.			6	6
14.B.5	The test manager is informed in time about changes in planned and agreed delivery dates (test basis as well as test object).			4	4
14.B.6	In a periodic defects meeting (or analysis meeting), solutions to defects are discussed between representatives of the test team and of other parties involved.			4	4

14.B.7	Testing is involved in change control for judging the impact of change proposals on the test effort.			4	4
14.C	Communication within the organization about the quality of the test process (C)				
14.C.1	There is a periodic meeting in which propositions for improvement of the test methodology used and the test processes are discussed			3	3
14.C.2	Participants are representatives of the test teams and of the line department for testing			3	3
15	Reporting				
15.A	Defects (A)				
15.A.1	The defects found are reported periodically, divided into solved and unsolved defects			8	8
15.B	Progress (status of tests and products), activities (costs and time, milestones), defects with priorities (B)				
15.B.1	The defects are reported, divided into severity categories according to clear and objective norms			5	5
15.B.2	The progress of each test activity is reported periodically and in writing. Aspects reported on are: lead time, effort spent, which tests have been specified, what has been tested, what parts of the object performed correctly and what must still be tested			4	4
15.C	Risks and recommendations, substantiated with metrics (C)				
15.C.1	A quality judgment on the test object is made. The judgment is based on the acceptance criteria, if present, and related to the test strategy			5	5
15.C.2	Possible trends with respect to progress and quality are reported periodically and in writing			5	5
15.C.3	The reporting contains risks (for the customer) and recommendations			5	5
15.C.4	The quality judgment and the detected trends are substantiated with metrics (from defect administration and progress monitoring)			6	6
15.D	Recommendations have a Software Process Improvement character (D)				
15.D.1	Advice is given not only to the area of testing but also on other parts of the project			8	8
16	Defect management				
16.A	Internal defect management (A)				

16.A.1	The different stages of the life cycle of defects are administrated (up to and including retest)			7	7
16.A.2	The following items are recorded about the defect: unique number, person entering the defect, date, severity category, problem description, status indication			8	8
16.B	Extensive defect management with flexible reporting facilities (B)				
16.B.1	Defect data needed for later trend analyses is recorded in detail (test case, test, subsystem, priority (test blocking Y/N), program + version, test basis + version, cause (probable + definitive), all status transitions of the defect including dates, a description of the problem solution, (version of) test object in which the defect is solved, problem solver)			4	4
16.B.2	The administration lends itself to extensive reporting possibilities, which means that reports can be selected and sorted in different ways.			6	6
16.B.3	There is someone responsible for ensuring that defect administration is carried out properly and consistently.			7	7
16.C	Project defect management (C)				
16.C.1	The defect administration is used integrally in the project. The defects originate from the various disciplines, those who perform the solution add their solution to the administration themselves, and so on.			8	8
16.C.2	Authorizations ensure that each user of the administration can do only what he or she is allowed to do			8	8
17	Testware management				
17.A	Internal testware management (A)				
17.A.1	The testware (test cases, starting databases, etc.), test basis, test object, test documentation, and test guidelines are managed internally according to a described procedure, containing steps for delivery, registration, archiving and referring			6	6
17.A.2	The management comprises the relationships between the various parts (test basis, test object, testware, etc.)			5	5
17.A.3	Transfer to the test team takes place according to a standard procedure. The parts comprising a transfer should be known: which parts and versions of the test object,			8	8

	which (version of the) test basis, solved defects, and still unsolved defects, including those from the developer himself				
17.B	External management of test basis and test object (B)				
17.B.1	The test basis and the test object (usually design and software) are managed by the project according to a described procedure, with steps for delivery, registering, archiving and reference			6	6
17.B.2	Management contains the relationships between the various parts (test basis and test object)			6	6
17.B.3	The test team is informed about changes in the test basis or test object in a timely fashion			6	6
17.C	Reusable testware (C)				
17.C.1	(A selection, which is agreed beforehand, of) the test products are completed after the end of the test (= fully and up to date) and transferred to the maintenance organization, after which the transfer is formally agreed			3	3
17.C.2	The transferred test products are actually reused			3	3
17.D	Traceability system requirements to test cases (D)				
17.D.1	Each system requirement and specification is related to one or more test cases in a transparent way, and vice versa			3	3
17.D.2	These relations are traceable through separate versions (for example, system requirement A, version 1.0 is related to functional design B, version 1.3 is related to programs C and D, versions 2.5 and 2.7, and is related to test cases X to Z, version 1.4)			3	3
18	Test process management				
18.A	Planning and execution (A)				
18.A.1	Prior to the actual test activities, a test plan is formulated in which all activities to be performed are mentioned. For each activity there is an indication of the period in which it runs, the resources (people and means) required, and the products to be delivered			4	4
18.B	Planning, execution, monitoring, and adjusting (B)				
18.B.1	Monitoring of the execution of all planned activities takes place			7	7

18.B.2	Each activity is also monitored in terms of time and money			7	7
18.B.3	Deviations are documented			7	7
18.B.4	In case of deviations, adjustments are made, either by adjusting the plan, or by performing activities again according to the plan. The adjustment is substantiated			7	7
18.C	Monitoring and adjustment within the organization (C)				
18.C.1	At an organizational level, there is monitoring of the application of the methodology (methods, standards, techniques, and procedures) of the organization			6	6
18.C.2	Deviations are documented and reported to the test process			7	7
18.C.3	In the case of deviations the risks are analyzed and adjustments are made, for example by adapting the methodology or by adapting activities or products so that they still meet the methodology. The adjustment is substantiated			7	7
19	Evaluation				
19.A	Evaluation techniques (A)				
19.A.1	In evaluating (intermediate) products techniques are used; in other words, a formal and described working method is applied			4	4
19.A.2	The evaluation and its results are reported			7	7
19.A.3	Then handling of the results is monitored			7	7
19.A.4	Testers are involved in these evaluations			7	7
19.B	Evaluation strategy (B)				
19.B.1	A conscious consideration of the products risks takes place			3	3
19.B.2	There is a differentiation in the area of consideration and the depth of the evaluations, depending on the risks taken and, if present, the acceptance criteria: not all (parts of) intermediate products are evaluated equally; this also goes for quality characteristics			3	3
19.B.3	Choices are made from multiple evaluation techniques, suitable for the desired depth of an evaluation			3	3

19.B.4	For re-evaluation a (simple) strategy determination takes place, in which a conscious choice of variations between 'evaluate solutions only' and 'complete re-evaluation' is made			3	3
19.B.5	The strategy is determined and subsequently executed. It is checked that the execution of the evaluations takes place according to the strategy; if necessary, adjustments are made			3	3
20	Low-level testing				
20.A	Low-level test life-cycle model (planning, specification, and execution (A)				
20.A.1	For the low-level test (at least) the following phase are recognized: planning, specification, and execution. These are performed in sequence, for each subsystem, if applicable				?
20.A.2	Activities to be performed for the planning phase are: formulate assignment, determine the test basis, set up organization, set up test products, define infrastructure and tools, set up management, determine planning, produce and agree test plan				?
20.A.3	Activities to be performed for the specification phase are: design test cases and test scripts				?
20.A.4	Activities to be performed for the execution phase are: set up starting test databases, execute (re)tests				?
20.B	White-box techniques (B)				
20.B.1	Besides informal techniques, low-level tests use also formal techniques, providing an unambiguous route from the test basis to test cases				?
20.B.2	For low-level tests it is possible to make a substantiated statement about the level of coverage of the test set (compared to the test basis)				?
20.B.3	The testware is reusable (within the test team) by a uniform working method				?
20.C	Low-level test strategy (C)				
20.C.1	A motivated consideration of the product risks takes place, for which knowledge of the system, its use and its operational management is required			4	4

20.C.2	There is a differentiation with respect to the area of consideration and the depth of the tests, depending on the risks taken and, if present, the acceptance criteria: not all kinds of programs are tested equally thoroughly: this is also the case for quality characteristics			4	4
20.C.3	One or multiple formal or informal test specification techniques are used, suitable for the desired test depth			5	5
20.C.4	For retests a (simple) strategy determination also takes place, in which a substantiated choice is made between variations of 'test solutions only' and 'complete retest'			5	5
20.C.5	The strategy is determined and subsequently executed. It is checked that the execution of the tests takes place according to the strategy; if necessary, adjustments are made			5	5

Anexo D – Procesos de los dominios del área de calidad

En el siguiente anexo se detallan cada uno los sub-procesos definidos en el área de calidad según los dominios cmmi. En ellos se presenta: Descripción, artefactos de entrada / salida, acciones o tareas y los roles involucrados.

Proceso: Recepción de documentación			Dominio: Planear y organizar
Descripción del proceso	En este proceso se realiza la recepción y análiss de la documentación por parte de las áreas que integran el desarrollo de software (estos documentos puede proveenir de otras sedes de la empresa)		
Entradas	- Documento de especificaciones funcionales – FSD. - Documento de pruebas funcionales. - Documento oficial de cambios, información técnica de cambios – ITF - Mapping changes. - Template changes.	Salidas	Presentación de cambios
Acciones	- Solicitar los documentos a cada una de las áreas correspondientes. - Organizar la documentación en el repositorio correspondiente. -Generar presentación de cambios y socializar con los integrantes del área de calidad y el jefe de proyectos asignado. - Definir los tipos de pruebas de alto nivel que se debería ejecutar sobre los cambios presentados.		
Roles	Jefe calidad, analistas de calidad, jefe de proyectos.		

Tabla 11. Detalle proceso recepción de documentación

Proceso: Definición y documentación de pruebas			Dominio: Planificación y análisis
Descripción del proceso	En este proceso se definirán el tipo de pruebas que deben ser llevadas a cabo según los componentes que tengan cambios o novedades. Los tipos de pruebas contemplados son: Pruebas de humo, pruebas de esfuerzo, pruebas de regresión y pruebas funcionales. Para la documentación de pruebas se hará uso de la herramienta Testlink		
Entradas	Presentación de cambios	Salidas	Casos de prueba
Acciones	- Especificar de los tipos de pruebas. - Definir la cobertura de las pruebas - Documentar las pruebas en la herramienta - Validar las pruebas con el área de Consultores de Negocio.		
Roles	Analista de calidad, interacción: consultor de negocio		

Tabla 12. Detalle proceso definición y documentación de pruebas

Proceso: Estimación de esfuerzo			Dominio: Planificación y análisis
Descripción del proceso	Definir las horas hombres de esfuerzo que tomará la ejecución de pruebas de aseguramiento de calidad sobre los cambios en el sistema.		
Entradas	- Presentación de cambios - Casos de Prueba	Salidas	- Horas hombre estimadas de esfuerzo
Acciones	- El jefe del área de calidad debe realizar la estimación de esfuerzo en base a indicadores registrados en la herramienta de control de tiempos. (La estimación podrá ser llevada a cabo por los analistas de calidad, si es solicitado por el jefe de área) - Se debe reportar la estimación al jefe de proyectos.		
Roles	Jefe de calidad, jefe de proyectos		

Tabla 13. Detalle proceso estimación de esfuerzo

Proceso: Administración de ambientes			Dominio: Planificación y análisis
Descripción del proceso	Administrados los distintos ambientes necesarios para el aseguramiento de calidad según las pruebas definidas.		
Entradas	Casos de prueba	Salidas	Correo de solicitud de ambientes Ambientes asignados al área de calidad.

Acciones	- Definir los tipos de ambientes necesarios para el área de calidad. - Solicitar dar de baja ambientes obsoletos - Solicitud de ambientes copia de producción.
Roles	Jefe de calidad, analistas de calidad

Tabla 14. Detalle proceso administración de ambientes

Proceso: Chequeo de recepción de cambios			Dominio: Ejecución y resultados
Descripción del proceso	Revisión que los cambios entregados e instalados por parte del área técnica correspondan a los requerimientos solicitados y cumplan con la documentación necesaria para el aseguramiento de calidad; aprobar la configuración inicial de los sistemas previo a la ejecución de pruebas		
Entradas	- Documento de especificaciones funcionales – FSD. - Documento de pruebas funcionales. - Documento oficial de cambios, información técnica de cambios – ITF - Mapping changes. - Template changes.	Salidas	Correo de aprobación
Acciones	- Revisión de la documentación final entregada por cada área involucrada en el proyecto. - Solicitud de aclaración sobre cambios. - Revisar instalación de todos los componentes generados. - Revisión de configuración necesaria para la puestas en marcha de los cambios o nueva versión. - Ejecutar pruebas de humos, para confirmar integración de cambios entre sistemas (de ser necesario).		
Roles	Analista de calidad, interacción con desarrolladores		

Tabla 15. Detalle proceso chequeo de recepción de cambios

Proceso: Ejecución de pruebas			Dominio: Ejecución y resultados
Descripción del proceso	Ejecución de pruebas para el aseguramiento de calidad de los sistemas		
Entradas	- Casos de prueba Apoyo: - Documento de especificaciones funcionales – FSD. - Documento de pruebas funcionales. - Documento oficial de cambios, información técnica de cambios – ITF - Mapping changes. - Template changes.	Salidas	Componentes de software validados
Acciones	- Ejecución de las pruebas de calidad correspondientes. - Documentación de evidencia del resultado de las pruebas sobre la herramienta testlink. - Reporte de avance semanal sobre la ejecución.		
Roles	Analistas de calidad		

Tabla 16. Detalle proceso ejecución de pruebas

Proceso: Documentación de casos de pruebas			Dominio: Automatización de pruebas
Descripción del proceso	Documentar los casos de prueba (de ser necesarios) o actualizar los casos en la herramientas testlink.		
Entradas	Listado de casos de pruebas	Salidas	Casos de pruebas
Acciones	- Determinar si los casos de prueba seleccionados ya se encuentran documentos. - Documentar o actualizar los casos de prueba.		
Roles	Analistas de calidad		

Tabla 17. Detalle proceso documentación de casos de pruebas

Proceso: Reporte de incidentes			Dominio: Ejecución y resultados
Descripción del proceso	Generar incidencias cuando se encuentren errores sobre el sistema, ya sean producidos por los cambios o preexistentes en los sistemas.		
Entradas	- Artefactos de software - Ambientes de pruebas - Casos de prueba	Salidas	- Incidencia reportada
Acciones	- Documentar incidencia en template. - Crear incidencia en la herramienta SBM. - Seguimiento de incidencias ajustadas durante el desarrollo del proyecto. - Reporte semanal de incidencias encontradas. - Relación de código de incidencia con caso de pruebas.		
Roles	Jefe de calidad, analistas de calidad		

Tabla 18. Detalle proceso reporte de incidentes

Proceso: Liberación de cambios			Dominio: Ejecución y resultados
Descripción del proceso	Informar a los involucrados la culminación de pruebas y aprobación final de los cambios.		
Entradas	- Artefactos de software - Incidencias	Salidas	- Correo de liberación de versión.
Acciones	- Recopilación final de la documentación anexa a la versión. - Reporte final de indicadores. - Informe final por correo de disposición de los cambios a los clientes.		
Roles	Analista de calida		

Tabla 19. Detalle proceso liberación de cambios

Proceso: Definición de cobertura			Dominio: Automatización de pruebas
Descripción del proceso	Definir los tipos de pruebas y los casos de pruebas que serán adaptados al sistema de automatización de pruebas.		
Entradas		Salidas	- Listado de casos de pruebas
Acciones	- Realizar la selección de las pruebas que deben ser automatizadas algunos de los criterios que se deben tener en		

	cuenta para la selección de casos son: ejecución repetitiva, criticidad de la funcionalidad, funcionalidad implementada en diferentes clientes, etc.
Roles	Jefe de calidad, analistas de calidad

Tabla 20. Detalle de proceso definición de cobertura

Proceso: Programación y estabilización de los casos			Dominio: Automatización de pruebas
Descripción del proceso	Creación de los scripts de pruebas, ejecución de los casos automatizados y comprobación de las validaciones		
Entradas	- Casos de pruebas (testlink)	Salidas	- Casos de pruebas automatizados
Acciones	- Automatización de casos de pruebas, según la arquitectura definida. - Ejecución de casos con revisión por parte de un analista de calidad para comprobar la cobertura del caso. - Ejecución repetitiva de los casos para avalar su estabilidad. - Actualización de casos según cambios que lo requieran.		
Roles	Desarrollador, analista de calidad		

Tabla 21. Detalle proceso programación y estabilización de los casos