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

AUTOMATIZACIÓN DE PRUEBAS DE REGRESIÓN

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

CRISTIAN ALEJANDRO RIVERA MARTÍNEZ

PROFESOR GUÍA:
JOCELYN SIMMONDS WAGEMANN

MIEMBROS DE LA COMISIÓN:
JORGE PEREZ ROJAS
ERIC TANTER
MARCELO MENDOZA ROCHA

SANTIAGO DE CHILE
2018

RESUMEN

En el desarrollo de software, existen tres elementos determinantes para la obtención de sistemas de calidad, estos son: las personas, la tecnología y los procesos, teniendo los procesos una incidencia significativa en la calidad del producto. En el proceso de *testing*, las pruebas de software permiten detectar fallas antes que los sistemas sean instalados en ambientes productivos. Los sistemas son cada vez más complejos, por lo cual, la automatización de pruebas de software es una estrategia que se está utilizando hoy en día en muchas empresas.

En la compañía de TV en la cual se desarrolló este trabajo de tesis, el proceso de *testing* es ejecutado manualmente y algunos de los problemas identificados en este proceso fueron: identificación tardía de errores producto de pruebas manuales; utilización de personas de otras áreas de la compañía en funciones de *testing* y el desarrollo de requerimientos con plazos limitados que restringen al área de tecnología de información el poder realizar las pruebas de regresión a funcionalidades previas en un 100% mediante las pruebas manuales.

Es por ello que en este proyecto de tesis se elaboró una herramienta de automatización de la ejecución de pruebas de regresión, que fue construida a partir de 2 prototipos. La implementación de esta herramienta dio como resultado la formalización de un nuevo proceso de *testing* que permite cubrir un mayor porcentaje de pruebas, reducir los tiempos de validación de los sistemas testeados, y asegurar la mantenibilidad de los *scripts* de prueba automatizados. Además, se logra una disminución de horas de trabajo asociadas a mantenciones correctivas y pruebas funcionales. Esta herramienta fue validada por un grupo de personas que participo en la ejecución de los prototipos.

DEDICATORIA

Mi tesis la dedico a mis queridos hijos Matías y Esteban, por ser ellos mi fuente de motivación para superarme cada día.

Especialmente a mí querida esposa Rossana, por su sacrificio y esfuerzo durante todos estos años, por motivarme y creer en mis capacidades. Muchas gracias amor.

AGRADECIMIENTOS

A mis colegas y jefe cuya colaboración fue muy importante en el desarrollo y posterior implementación de esta iniciativa al interior de la organización.

A la universidad y profesores, por la calidad en su enseñanza y el entusiasmo que brindan día a día en la formación profesional de todos sus alumnos.

A la profesora Jocelyn por aceptar mi iniciativa y por guiarme durante muchos meses en la construcción de esta tesis.

Muchas gracias a todos.

TABLA DE CONTENIDO

ÍNDICE DE IMÁGENES

1. INTRODUCCIÓN

El presente capítulo describe a un alto nivel a la empresa y el área en que se desarrolló el trabajo de tesis, proporcionando información de la estructura que permita entender el contexto organizacional en la que se desenvuelve el problema a resolver, cuáles fueron los objetivos planteados y la motivación personal para llevarlo a cabo. Se incluye también la descripción del contenido de cada uno de los capítulos permitiendo entender la organización del documento.

1.1. Contexto de la empresa

Debido a políticas de confidencialidad y seguridad el nombre de la empresa no puede ser indicado. Por lo cual en el presente documento de tesis será mencionada como la compañía de TV.

La compañía de TV es un proveedor de televisión satelital que brinda experiencia de televisión a través de la adquisición, producción y distribución de contenido exclusivo y único, y el despliegue continuo de las últimas tecnologías para el entretenimiento en televisión. En Latinoamérica, esta compañía provee servicio en más de 10 territorios, incluyendo Brasil, Argentina, Chile, Colombia, Ecuador, México, Perú, Puerto Rico, Uruguay, Venezuela y el Caribe.

El proyecto de tesis se desarrolló en la filial chilena de la compañía, dentro del área de tecnología de la información (TI), específicamente enfocado en el proceso de *testing* de los sistemas desarrollados por la compañía de TV.

En la imagen 1, se visualiza la estructura organizacional de TI en Chile, liderada por el gerente senior de tecnología de información. En esta estructura organizacional se visualizan 2 gerencias en TI, la Gerencia *Solution Delivery* (SD), tiene como responsabilidad la administración del software de todos los sistemas utilizados por la compañía de TV y la Gerencia S*ervice Delivery Manager* (SDM), tiene como responsabilidad la administración de la plataforma operacional (hardware, redes y comunicación).

En la estructura organizacional, los centros de competencias representan a grupos de personas con conocimiento técnico especializado en sistemas que son utilizados en varios países de Latinoamérica, siendo estos los siguientes sistemas: El sistema CORE, el sistema de ventas, el sistema de inteligencia de negocio (BI), y el sistema de operación en terreno de los técnicos. En la gerencia SD, el equipo

de calidad es responsable del soporte nivel 2[1] de todos los sistemas y del proceso de *testing*. El equipo de operaciones es responsable de la mantenibilidad a los sistemas de instalación y logístico. Finalmente, el equipo de backoffice es responsable de los sistemas de recaudación y facturación.

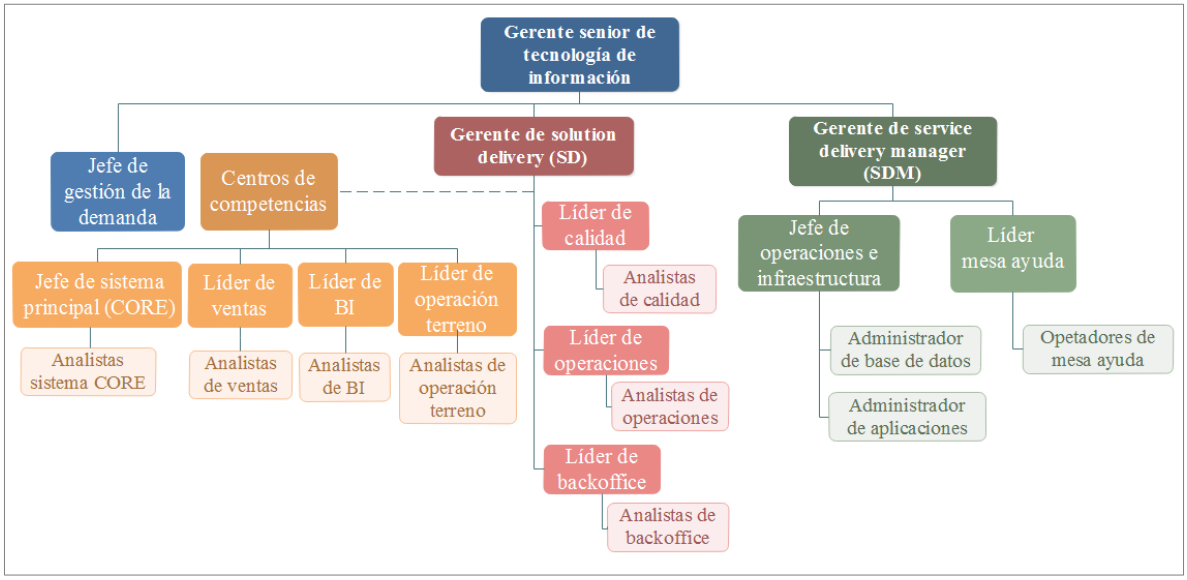

Imagen 1: Estructura organizacional de TI.

Los sistemas antes mencionados tienen como principal característica que fueron construidos mayoritariamente por personal de TI de la compañía de TV, y en una arquitectura que permite ser operados desde los navegadores web de las estaciones de trabajo.

Por normativa y procedimiento de auditoria se exige al área de TI cumplir con un checklist de aprobación del usuario funcional para cualquier paso a producción de un nuevo release de los sistemas existentes en el site[2] de sistemas. La aprobación se obtiene mediante la ejecución de pruebas de aceptación del usuario (por sus siglas en ingles es llamada *User Acceptance Testing* (UAT)) y la responsabilidad de asegurar la aprobación del checklist, recae sobre el equipo de calidad de la gerencia de SD. Generalmente para poder dar más velocidad y capacidad al

[1] El soporte nivel 2 atiende los tickets asociados a incidentes en los sistemas y que fueron ingresados por los usuarios de la compañía de TV que no pudieron ser resueltos en una primera línea por los operadores de la mesa de ayuda.

[2] Representa a los servidores ubicados en una sala denominada “SITE”. Esta sala se encuentra equipada para mantener una temperatura baja (entre 21 a 23 grados Celsius) y la cual contiene a todos los sistemas de la compañía de TV de Chile (CORE, ventas, atención clientes, entre otros).

proceso de *testing*, se solicitan usuarios a las otras áreas de la compañía para apoyar las tareas de pruebas del sistema.

1.2. Motivación para la realización del proyecto

La búsqueda de productividad es un desafío importante que está presente en muchas organizaciones y no es la excepción para la compañía de TV Chile. Año tras año se plantea en las distintas áreas de la compañía el desafío de mejorar la productividad con eficiencia, reduciendo presupuestos sin afectar el servicio proporcionado. Existen muchas alternativas a la hora enfrentar la eficiencia en la productividad, una de ellas es revisar las formas de trabajo para optar por alternativas y en este contexto se identificó la realización de las pruebas de forma manual en el área de TI de los sistemas como una oportunidad de mejora, ejecutándose las mismas pruebas de validación en los 3 entornos de trabajo (desarrollo, *testing* y pre-producción).

El *testing* ejecutado de forma manual tarda entre 3 a 4 días en cada uno de estos entornos y de no existir error invalidante que implicaría más días para la validación completa, por lo general demora 2 semanas las pruebas de un nuevo release.

La motivación para este proyecto surge de la necesidad de poder reducir el tiempo y esfuerzo que requiere el proceso de pruebas manuales, y que permita al equipo participante de las pruebas realizar ciclos de pruebas que no sean costoso en tiempo como viene siendo.

1.3. Problema a resolver

En el contexto del desarrollo de software, las pruebas de validación buscan comprobar que el software hace lo que el usuario espera, siendo éstas una parte importante del proceso de desarrollo, que requiere de un gran compromiso por parte de todo el equipo involucrado. Sin embargo, el proceso de *testing* manual es tedioso una vez que una prueba ha sido ejecutada en más de una ocasión. Hablando del caso específico del proceso actual de *testing* del área de sistemas de la compañía de TV tiene problemas que se resumen en los siguientes puntos:

1. Las pruebas manuales de regresión involucran más tiempo del que se dispone en la compañía de TV para la validación de cada nuevo release, donde la cantidad de pruebas ejecutadas es restringida solo a un porcentaje de ellas por limitaciones de tiempo. Además, por ser validaciones manuales pueden inducir a pruebas mal ejecutadas.
2. La identificación de los errores es tardía debido a que la etapa de pruebas se realiza al final del ciclo de desarrollo, una o dos semanas antes al paso a producción, lo que en algunos casos implica entregar un nuevo release con defectos y también, una importante utilización de horas para mantenciones correctivas que se podrían destinar a otros requerimientos evolutivos.

3. Se utilizan personas de otras áreas de la compañía para la ejecución de pruebas de regresión, producto de la limitación de tiempo y personas del equipo de calidad de TI. Lo anterior, también afecta el cumplimiento de las metas internas de las otras áreas al ceder parte de su personal para este tipo de actividad.
4. Por último, los casos de prueba de regresión ejecutados de forma manual no cubren el 100% de las funcionalidades de los sistemas y tampoco considera escenarios de funcionalidades en los que los sistemas no debiesen poder realizar, impidiendo descubrir posibles defectos adicionales.

1.4. Objetivos del trabajo

Tanto el objetivo principal y los objetivos específicos que se indican están muy relacionado con poder abordar mejoras en la productividad planteados en la motivación personal antes descrita.

El objetivo principal de este trabajo es poder definir, formalizar e instrumentar un proceso de *testing* enfocado en la automatización de las pruebas de regresión que resulte beneficioso en tiempo y esfuerzo en comparación con la ejecución manual de las pruebas. En este trabajo de tesis, se trabajará con los sistemas de ventas y el sistema de atención a clientes, para que sirva como referencia a otros sistemas para entregar software de mejor calidad en la compañía de TV Chile.

Al momento de iniciar el proyecto, estos 2 sistemas seleccionados como piloto, eran los que más horas de mantención correctiva ocupaban producto de los errores en ambiente de producción y también, los que requerían de un volumen importante de pruebas de regresión que no se alcanzaba a ejecutar de forma manual.

1.5. Objetivos específicos

Los objetivos específicos que se desprenden a partir del trabajo a realizar son:

1. Reducir el tiempo que transcurre entre que un requerimiento es concebido y está disponible en el sistema para ser usado en producción. Esta métrica se conoce como time to market (TTM), donde la definición de un nuevo proceso de *testing* y la realización de las pruebas automatizadas de regresión permitirá a los equipos de SD realizar validaciones en menor tiempo y repetirlas según sea necesario, haciendo posible además, desarrollar más requerimientos evolutivos en un mismo release en los sistemas de ventas y el sistema de atención de clientes. La métrica time to market actualmente es 90 días y el objetivo es reducirla a 65 días.
2. Reducir las horas hombre (HH) invertidas en pruebas manuales por los equipos de SD, donde la realización de pruebas automatizadas de regresión en los sistemas antes descritos, permitirá realizar una mayor cobertura de pruebas

utilizando menos tiempo que un proceso manual. Actualmente se destina para las pruebas manuales el 30% del tiempo total por requerimiento y el objetivo es reducirlo al 10%.

3. Reducir los casos de prueba que se ejecutan de forma manual, lo que implica automatizar más del 65% de los casos de prueba de cada sistema. En el caso del sistema de ventas, la cantidad actual de pruebas manuales definidas son 70 y el objetivo es automatizar 50 casos de prueba. En el caso del sistema de atención de clientes, la cantidad actual de pruebas definidas son 296 y el objetivo es automatizar 200 casos de prueba.
4. Reducir las HH para pruebas UAT utilizando a los usuarios de negocio, donde la validación funcional considerará sólo realizar pruebas manuales de las nuevas funcionalidades de los releases. En la actualidad se requieren al menos 4 usuarios por cada nuevo release y el objetivo es disminuir a la mitad el número de usuarios a solicitar a las otras áreas de la compañía para la ejecución de las pruebas.
5. Reducir las HH invertidas para mantenciones correctivas, donde la realización de pruebas automatizadas de regresión, permitirá realizar una mayor cobertura de pruebas y detectar en los ambientes no productivos fallos introducidos producto de las nuevos requerimientos en las funcionalidades previas. Se medirá con los indicadores que se presentan en la reunión mensual de TI. En esta reunión participan los gerentes, jefes y líderes de la dirección de sistemas. En la actualidad se utiliza el 43% del total de HH disponibles de SD en mantenciones correctivas y el objetivo es reducirlo al 30%.

1.6. Capítulos de la tesis

En esta sección se presenta brevemente el contenido de los restantes capítulos que componen esta tesis.

En el capítulo 2, se explican conceptos básicos que fueron utilizados para formular el trabajo de tesis. Se presentan los distintos tipos de prueba y se proporcionan ejemplos hasta llegar a la definición de las pruebas de regresión, casos de prueba, *script* de prueba y de *script* automatizados. Se finaliza este capítulo detallando algunos métodos de *testing* y de trabajos relacionados al marco en que se desenvuelve esta tesis.

En el capítulo 3, se describe todo el marco sobre el que se ha estado ejecutando las pruebas manuales en la compañía, entregando información del contexto funcional, tecnológico de los sistemas y que permita al lector comprender desde los problemas, la solución realizada.

En el capítulo 4, se describe un primer prototipo, mostrando cada una de las etapas de desarrollo de software, estas son: análisis, diseño, construcción y transición. Se finaliza este capítulo con algunas conclusiones preliminares que se obtuvieron puntualmente respecto al prototipo.

En el capítulo 5, se describe la implementación de un segundo prototipo, en el que se incluyó mejoras de algunas funcionalidades respecto del primer prototipo, las que permitieron alcanzar mejoras en la productividad que fue planteada en la motivación de la tesis.

En el capítulo 6, se presenta los resultados de la implantación de ambos prototipos, analizando el cumplimiento del objetivo principal y los objetivos específicos que se declararon.

Finalmente en el capítulo 7, se presenta las conclusiones finales del trabajo.

2. MARCO TEORICO

En este capítulo se definen algunos de los términos que fueron considerados importantes para la formulación del trabajo de tesis respecto del alcance que estos representan en el marco teórico, algunos de ellos son: estrategias de prueba, casos de prueba, automatización de pruebas, métodos para las pruebas automatizadas, el enfoque de prueba centrada en los datos y por último, se mencionan algunos trabajos existentes acerca de automatización de pruebas.

2.1. Estrategias de prueba

Una estrategia de pruebas, no es una serie de instrucciones que detallan el paso a paso de cómo se debe hacer las pruebas. La estrategia define los objetivos a alcanzar, a menudo con recursos limitados y en condiciones de incertidumbre, donde se construye un plan de alto nivel para lograr los objetivos específicos de la prueba. La estrategia que se adopte dependerá de principalmente del proceso de desarrollo que se utiliza al interior de una organización TI y de las capacidades existentes del equipo de prueba [Copeland, 2015].

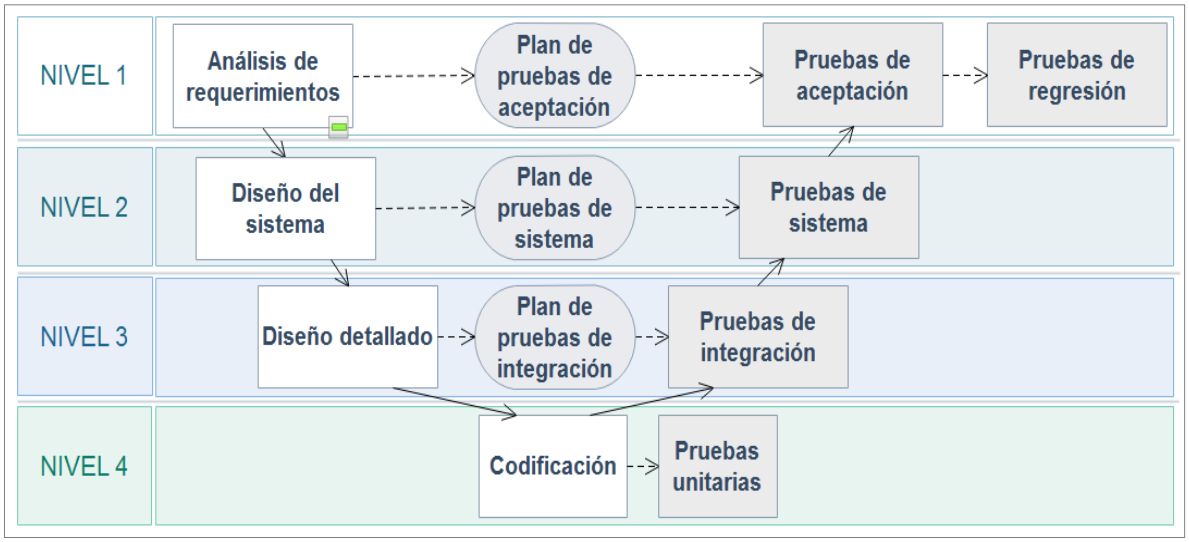

Imagen 2: Estrategias de pruebas según fases del proceso desarrollo.

El modelo en V, o también conocido como 4 niveles, es uno de los modelos de validación de pruebas existentes para el desarrollo del software, cuya estrategia está representada en la imagen 2, propone para cada fase del desarrollo, una fase paralela de verificación o validación. Esta estructura obedece al principio: para cada fase del desarrollo debe existir un resultado verificable [Soria, 2016].

En la parte izquierda de la imagen 2, representa el flujo donde se desarrolla el sistema y la parte derecha, representa el flujo donde se comprueba el sistema (contra las especificaciones definidas en la parte izquierda).

En los próximos párrafos se describirán algunos tipos de pruebas que se recomienda realizar de acuerdo a la prueba deseada según el nivel, las que se encuentran relacionadas a las fases del proceso de desarrollo.

Las pruebas unitarias (nivel 4), están limitadas en la unidad más pequeña de software, la componente y el tipo de prueba recomendada es caja blanca o estructural. Por ejemplo: si se construye un método “sumar(x,y)” que permite sumar 2 números como parte de la funcionalidades de un sistema y se requiere validar específicamente este método, se crea otra rutina de código para validarla, como “validarSuma(1,2)=3”, que debe ser ejecutada cada vez que se considere necesario.

Para las pruebas de integración (nivel 3), donde requiere integrar a más de un componente del software, ya en una etapa más avanzada del desarrollo, se recomienda usar un enfoque de tipo *top-down* o *bottom-up*.

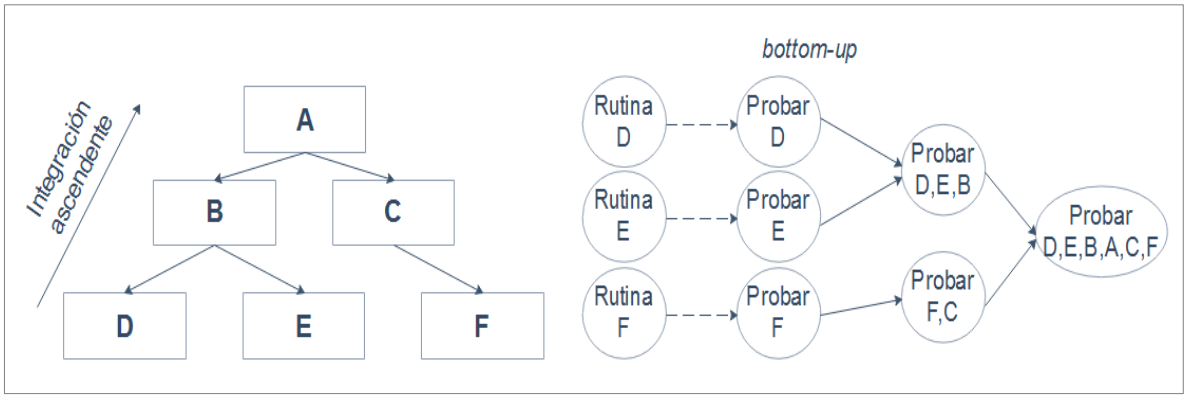

Imagen 3: Enfoque bottom-up de *testing*.

Usando una estrategia *bottom-up*, se prueban primero los componentes de bajo nivel y luego se combinan según jerarquía con los de mayor nivel. En *top-down*, el enfoque es exactamente inverso. Por ejemplo, si se tiene los componentes (A, B, C, D, E, F) con la jerarquía que se presenta a la izquierda de la imagen 3, utilizando un enfoque de pruebas *bottom-up,* se comienza probando con pruebas unitarias los componentes D, E y F, y posteriormente los componentes integrados al de mayor jerarquía (D, E, B) y (F,C), mediante rutinas codificadas especialmente para invocar a la combinación de componentes que se muestra a la derecha de la imagen 3, permitiendo así realizar las pruebas de integración. En contraste, en *top-down,* la componente principal A se prueba aisladamente,

después todos los componentes llamados por la componente A se combinan y se prueban como una unidad mayor.

Tradicionalmente las pruebas de sistema (nivel 2), las pruebas de aceptación de usuarios (nivel 1) y las pruebas de regresión (nivel 1) se realizan a partir de la definición de casos de prueba, estos pueden ser definidos como:

> "Un caso de prueba es (o debería ser) un producto de trabajo reconocido. Un caso de prueba completo contendrá un identificador de caso de prueba, una breve declaración de propósito (por ejemplo, una regla comercial), una descripción de condiciones previas, las entradas de casos de prueba, los resultados esperados, una descripción de las post condiciones esperadas y un historial de ejecución", [Jorgensen, 2013].

Para demostrar que el sistema satisface los requerimientos solicitados por el cliente, se utiliza las pruebas de aceptación de usuario (nivel 1). Se requiere al menos un caso de prueba por cada nuevo requerimiento solicitado.

En etapas posteriores, cuando el sistema ya está en uso y se realizan nuevos requerimientos que se integran al sistema, se recomienda realizar pruebas de regresión (nivel 1), donde se puede definir estas pruebas de acuerdo a lo siguiente:

> "Una prueba de regresión es una nueva prueba de un programa previamente probado, a realizarse después de la modificación de este; para garantizar que no se hayan introducido o descubierto fallos como resultado de los cambios realizados (descubrimiento de defectos enmascarados)", [Spillner, 2014]

Por ejemplo, en la tabla 1 se describe el caso de prueba (CP) para acceder al sistema de ventas.

Nombre	CP 01. Acceder al sistema.
Precondición	No aplica.
Pasos de ejecución (secuencias)	1. Escribir en el browser la siguiente url: http://sistemas.ventas.cl. 2. Ingresar usuario y contraseña en la posición solicitada. (Usuario=cmilthon01, Contraseña=CP01Cont). 3. Hacer *click* en botón aceptar.
Alternativas	No aplica.
Errores	1. El sistema indica mensaje "usuario incorrecto". 2. El sistema indica mensaje "contraseña incorrecta". 3. El sistema proporciona acceso a perfil erróneo de

	usuario ingresado.
Postcondición	1. El sistema debe mostrar solo el menú correspondiente al módulo de venta. 2. En el caso de mal ingreso de usuario o contraseña, debe aparecer el mensaje “Usuario o contraseña incorrecta”. 3. En el caso que exista problemas de conexión interna del sistema, debe enviarse el mensaje “comuníquese con mesa de ayuda, en caso de tener problemas de conexión”.

Tabla 1: Descripción de un caso de prueba.

En las pruebas de regresión, se recomienda diseñar los casos de prueba que se van a ejecutar cubriendo todas las funcionalidades de la aplicación, considerando 2 aspectos; la aplicación hace lo que se supone que hace (testeo positivo) y también, la aplicación no hace lo que no debe hacer (testeo negativo). Incluso se debería poder agregar aquellos casos de prueba generados producto de un incidente, estos últimos deben ser fácilmente identificables con el nombre que se asigna al CP [Graham, 2012].

2.2. Automatización de pruebas

El primer concepto relacionado al *testing* automatizado es *script* de prueba, el cual representa el paso a paso que se debe ejecutar de forma manual para validar una condición del sistema. Cuando se transforman estos *scripts* de prueba, en *scripts* automatizados utilizando herramientas especializadas aparece el concepto de *testing* automatizado y es usado generalmente para automatizar las pruebas de regresión [McKay, 2014].

Cuando estos *scripts* automatizados son utilizados en una iteración de un proceso de prueba, considerando además, los datos que se utilizarán para la realización de éstas, aparece el término de *test* de ejecución.

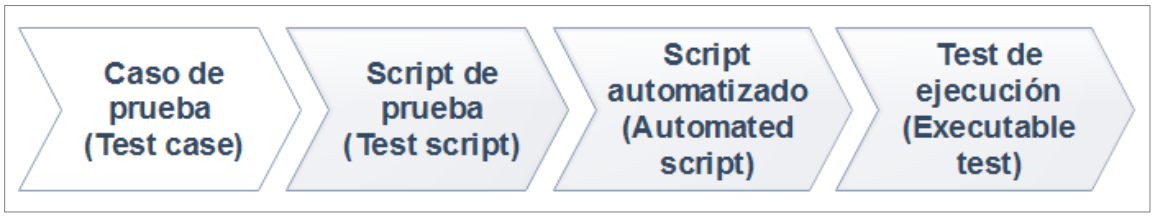

Imagen 4: Transformaciones de un CP en test de ejecución.

La transformación de los *scripts,* que se visualiza en la imagen 4, es lo que se considera tradicionalmente como automatización en las pruebas de sistemas.

Se puede destacar como principal beneficio al realizar la automatización de las pruebas de regresión: la reducción del tiempo de duración y esfuerzo en la ejecución, posibilidad de repetición automática y amplitud de cobertura, permitiendo entregar software de mayor calidad en comparación con las pruebas manuales [Madi, 2013].

2.3. Métodos de *testing* automatizado

Las herramientas de automatización no son suficientes para realizar la automatización de pruebas, también es necesario considerar el método a usar para realizar correctamente el *testing* automatizado y así evitar realizar doble trabajo.

En esta sección se presentarán tres métodos que pudieran usarse en beneficio de una correcta automatización de pruebas: descomposición funcional, *keyword-driven* y *action-based testing*, cuyos enfoques permiten identificar elementos a considerar en la construcción de los *scripts* automatizados. Estos son: funcionalidades comunes en los casos de prueba de un sistema, pruebas estándares, formatos y reglas generales de cómo las pruebas se deben documentar y administrar, estas últimas tareas son muy importantes pero pocas veces se consideran en las automatizaciones de prueba [Yépez, 2004].

2.3.1. Método de descomposición funcional

Para construir un *script* de prueba con el método de descomposición funcional, se debe reducir todos los casos de prueba a sus tareas más esenciales y escribir *scripts* definidos por el usuario, *scripts* de funciones de negocio y *scripts* de rutinas que efectúan esas tareas independientes unas de las otras. Para lograr la descomposición es necesario separar los datos de las funciones. Esto permite escribir un *script* de prueba automatizado para una función, usando archivos de datos para proporcionar tanto el input y la verificación de los resultados esperados. Se emplea un diseño estructurado o modular [Hayes, 1996].

2.3.2. Método *keyword-driven*

Para construir un *script* de prueba con el método *keyword-driven,* los casos de prueba deben estar escritos en una hoja de cálculo la que contiene *keyword* y/o valores especiales que son leídas por los *scripts* automatizados, estas palabras claves tienen un significado especial y generan una acción. El proceso completo es totalmente dirigido por los datos [Zambelich, 1998].

2.3.3. Método *Action-based testing* (ABT)

La principal diferencia de este método en comparación de los anteriores es su evolución en cuanto a la abstracción en la escritura de los *scripts*. No se muestran

detalles sobre cómo sería ejecutada la prueba, cuáles serán los botones a presionar o qué campos se deben rellenar en los cuadros de diálogo. De existir acciones de verificación de estado durante la ejecución de la prueba, no necesita especificar dónde capturar el valor real de comparación. Este valor podría ser capturado desde una ventana, pero también podría ser recuperada de una base de datos con una consulta SQL.

En ABT, las pruebas están organizadas en módulos de prueba. En cada módulo de prueba, se escribe como una serie de acciones, compuesto de una palabra clave de la acción y argumentos que representan los datos de entrada o los resultados esperados. Esta forma de trabajo mejora la lectura de las pruebas y tiende a ahorrar enormemente la capacidad de mantenimiento debido a que los esfuerzos de automatización se centran exclusivamente en las palabras de acción [Buwalda, 2011].

2.4. Enfoque centrado en los datos

En todos los métodos presentados anteriormente se hace referencia en atención a la obtención de los datos. Encontramos entonces el enfoque centrado en los datos y corresponde al uso de los datos en los *test* de ejecución, usualmente en la forma simples archivos CSV (*Comma Separated Values*) de texto, para administrar el proceso automatizado de prueba.

El *testing* centrado en datos puede ser expandido para incluir tanto datos de control como datos de prueba. Los datos de prueba que se introducen pueden en algunos casos utilizarse para testear al mismo tiempo algunas reglas de validación de datos y la funcionalidad de la aplicación que se representa en la GUI.

Las ventajas de los *scripts* de prueba centrada en datos son: no es necesario modificar el *script* de prueba para considerar datos adicionales, debido a que estos se agregan al archivo de texto asociados al *script*; de la misma forma, es fácil modificar los registros de datos; y finalmente, se pueden desarrollar registros múltiples de datos de prueba como variantes funcionales.

El ejemplo de la imagen 5 y tabla 2 [Hayes, 1996], describe un *script* automatizado utilizando el enfoque centrado en los datos, el cual extrae desde un archivo los datos (representado por la tabla 2) que se utilizan durante la ejecución de la pruebas. El CP automatizado tiene como alcance poder desde el archivo suministrar las entradas - selecciones de menús, botones de radio, cuadros de lista, casillas de verificación, y los botones, así como texto y pulsaciones de teclas que fueron programadas y almacenadas en la codificación del *script*. Por ejemplo, Type ACCTNO (línea 7 de la imagen 5) introduce al programa el valor 100 del campo (fila 1, columna 2 de la tabla 2) en la primera recurrencia del archivo leído.

```
Select menu item "Chart of Accounts>>Enter Accounts"
Open file "CHTACCTS.TXT"              * Open test data file
Label "NEXT"                          * Branch point for next record
Read file "CHTACCTS.TXT"              * Read next record in file
End of file?                          * Check for end of file
   If yes, goto "END"                 * If last record, end test
Type ACCTNO                           * Enter data for account #
Press Tab
Type ACCTDESC                         * Enter data for description
Press Tab
Select Radio button STMTTYPE          * Select radio button for statement
Is HEADER = "H"?                      * Is account a header?
  If yes, Check Box HEADER on * If so, check header box
Select list box item ACCTTYPE * Select list box item for type
Push button "Accept"
Verify text MESSAGE                   * Verify message text
  If no, Call LOGERROR                * If verify fails, log error
   Press Esc                          * Clear any error condition
CALL LOGTEST                          * Log test case results
Goto "NEXT"                           * Read next record
Label "END"                           * End of test
```

Imagen 5: *Script* automatizado con enfoque centrado en los datos [Hayes, 1996].

Test Case	ACCT NO	SUB ACCT	ACCT DESC	STMT TYPE	ACCT TYPE	HEADER	MESSAGE
1000000	100	0000	Current Assets	Balance Sheet	Asset	H	Account Added
1001000	100	1000	Cash in Banks	Balance Sheet	Asset		Account Added

Tabla 2: Estructura del archivo de prueba del ejemplo [Hayes, 1996].

2.5. Trabajos relacionados

Al realizar una investigación para el desarrollo de este proyecto, se han encontrado muchas herramientas de soporte para los distintos aspectos relacionados a procesos de automatización de pruebas de software, según su función o enfoque las podemos clasificar en:

- Herramientas enfocadas al desarrollador (útiles para las pruebas unitarias).

- Herramientas para la administración de pruebas (útiles para gestionar las pruebas).
- Herramientas que permiten desarrollar las pruebas funcionales (útiles en las pruebas de regresión).
- Herramientas para aplicar pruebas de carga.
- Y por último, herramientas para el manejo de defectos.

Todas estas herramientas pueden funcionar de forma individual o integrada ejecutando las distintas funciones para las que fueron construidas.

En los siguientes párrafos de esta sección se discute únicamente las herramientas de *testing* funcional, que es el tipo de *testing* que se enfoca el proyecto de tesis.

Estas herramientas permiten revisar que las aplicaciones funcionan de acuerdo a lo esperado, proveen facilidades para captura y ejecución automática (reproducir la captura) permitiendo grabar una aplicación a nivel funcional (tradicionalmente se utilizan para grabar pruebas de regresión). Una condición importante cuando se elige la herramienta de automatización, es la plataforma de ejecución de las aplicaciones que se requiere automatizar y donde estas tienen posibilidad de alcance, siendo hoy en día la plataforma web, el más común donde se han enfocado estas herramientas.

En este contexto existen varias herramientas gratuitas que permiten realizar las pruebas funcionales, emular el usuario final y poseen interfaz de grabación, por nombrar algunas importantes: Sahi [Sahi, 2016], WatiN [WatiN, 2013] y Selenium [Selenium, 2016].

Por mencionar solo algunas características de estas herramientas, todas ellas permiten testear una aplicación web a través de cualquiera de los navegadores más comunes al día de hoy: IE, Firefox, Opera, Chrome y Safari. Se diferencian principalmente en el lenguaje de programación usado en la construcción del *script* automatizado. Sahi implementa los *scripts* en JavaScript, WatiN implementa los *scripts* en .Net y Selenium implementa los *scripts* en Java como lenguaje nativo y también en C#, Perl, PHP, Python y Ruby.

Al respecto, [Mendoza, 2011] realizó una evaluación de todas estas herramientas, asignándoles a los diferentes criterios de evaluación una puntuación entre 1 y 10 y a cada criterio un peso entre 1 y 100. La herramienta que obtuvo una ponderación mayor fue Selenium en los conceptos que se visualizan en la tabla 3. Esta ponderación fue muy útil para la selección de la herramienta a considerar para el proyecto de tesis, sobre todo en lo que representa la estabilidad y mantenimiento de los *scripts* y la facilidad de uso que se indica ya que no se cuenta con experiencias previas dentro de TI en materia de automatización.

Concepto	Peso	Sahi	Selenium	Watir
Documentación	5%	8	10	7.4
Soporte online	5%	8	7	5
Actualizaciones de software	5%	9	9	9
Compatibilidad con lenguajes de programación	10%	8	10	7.4
Estabilidad y mantenimiento de los scripts	30%	4.7	8	8
Organización de los tests	10%	7	8	4
Compatibilidad de la interfaz de grabación	5%	10	4	4
Calidad de la interfaz de grabación	5%	6.2	7.1	5.5
Ejecución	10%	5.5	10	3.5
Facilidad de uso	15%	7.2	8.6	7.2
	100%	6.3	8.345	6.515

Tabla 3: Resumen del análisis de comparación [Mendoza, 2011].

A continuación se proporciona más detalle solo de la herramienta Selenium.

Selenium está compuesta por diferentes herramientas o componentes de software, cada una con un enfoque diferente para el apoyo a la automatización de pruebas, proporcionando distintas soluciones para abordar los problemas de automatización.

Estas herramientas dan lugar a un amplio conjunto de funciones de prueba dirigidos específicamente a las necesidades de las pruebas de aplicaciones web de todo tipo, proporcionando operaciones muy flexibles para la localización de elementos de interfaz de usuario y la comparación de resultados de la prueba contra comportamiento de la aplicación real. Una de las características clave de Selenium es el apoyo para la ejecución de las pruebas en múltiples plataformas de navegadores.

La principal diferencia de Selenium y otras herramientas de prueba para aplicaciones en web, esta utiliza un navegador real impulsado a través de JavaScript para reproducir los *scripts* de pruebas. Esto significa que tiene algunas características únicas que no están disponibles en ninguna otra herramienta de prueba web [Gheorghiu, 2005].

Las herramientas o componentes de la suite Selenium son:

- Selenium IDE: Es un entorno de desarrollo integrado para *scripts* de Selenium. Se implementa como una extensión de Firefox por intermedio de un plugin que permite grabar, editar y reproducir las pruebas. Provee el entorno ideal para crear *scripts* automatizados de manera rápida.
- Selenium Remote Control (RC): Corresponde a la primera versión de Selenium que permitió la ejecución de *scripts* automatizados considerando una arquitectura cliente – servidor. Hoy esta versión de Selenium está oficialmente obsoleta, la nueva versión es Selenium WebDriver, sin embargo, como contexto, el servidor aceptaba comandos vía http desde el cliente. La principal ventaja de RC, permitía la ejecución de los *scripts* en varios lenguajes de programación (Java, C#, Perl, PHP, Python y Ruby) para una mejor integración de Selenium a entornos de prueba existentes.
- Selenium WebDriver: Es una API orientada a objeto que permite ser usada desde otras herramientas IDE de programación. La principal ventaja de esta herramienta es que permite construir *scripts* de automatización de manera escalable, extensible, modificables y reutilizables. Estas características impulsaron a esta herramienta a ser considerada como el sucesor de Selenium RC.

  Selenium WebDriver tiene componentes que permite la captura de páginas web dinámicas, donde los elementos de una página pueden cambiar, sin que la propia página se vuelva a cargar. El objetivo de WebDriver es suministrar una API orientada a objetos bien diseñado que proporciona un soporte mejorado para los modernos problemas avanzados de pruebas de aplicaciones web.
- Selenium Grid: permite ejecutar las pruebas en diferentes máquinas con diferentes navegadores en paralelo. Es decir, la ejecución de varias pruebas al mismo tiempo considerando distribuir la ejecución de las prueba.

A continuación solo se realiza un bosquejo de la arquitectura física y lógica de la herramienta Selenium IDE, que se utilizará en el capítulo 4 para una prueba conceptual.

- Arquitectura física de Selenium IDE

  Selenium IDE en la documentación formal de su sitio web, no muestra de forma explícita un diagrama de arquitectura que permita respaldar teóricamente el diseño generado. Sin embargo, de acuerdo a las instrucciones de instalación podemos definir que se trata de una arquitectura física monolítica mostrada en la imagen 6.

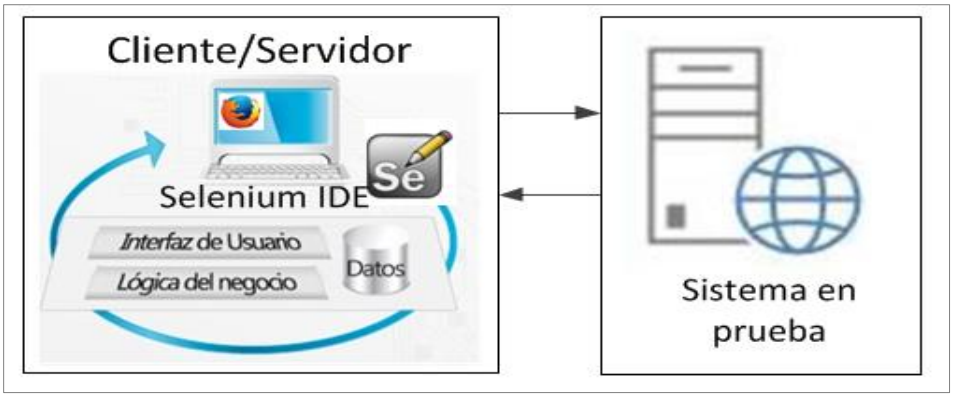

Imagen 6: Diagrama de arquitectura monolítica de Selenium IDE.

La arquitectura soporta múltiples usuarios ya que solo usa los recursos de la maquina cliente, permitiendo ejecutarse en varias instancias del navegador Firefox.

- Arquitectura lógica de Selenium IDE

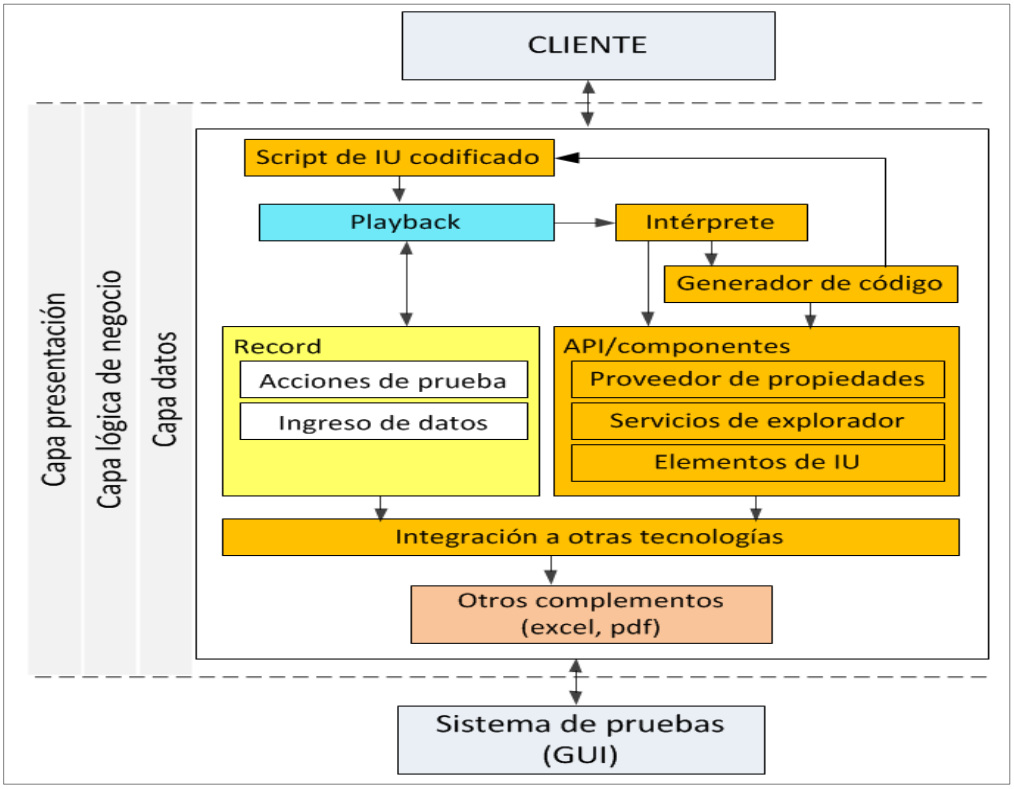

Imagen 7: Diagrama de arquitectura lógica de la aplicación Selenium.

Selenium IDE por ser una herramienta de entorno de desarrollo integrado y ejecutado en un 100% desde el cliente, contiene en un solo programa las 3 capas en el contexto de arquitectura lógica de sistemas multicapa: capa de presentación, capa de negocio y capa de datos. Como se visualiza en la imagen 7.

Dentro de los beneficios de esta arquitectura se encuentra el bajo costo que significa su implementación, opera en una misma máquina tanto el cliente como servidor, sin embargo, una desventaja importante, no permite aislar la lógica de un *script* de prueba en componentes separados y hacer del proceso de automatización de *testing* escasamente escalable, extensible y modificable. Es decir, no se podría separar la capa de presentación de una capa de lógica de negocio o de los datos.

En varios casos de estudio citados por [Graham, 2012], se indica a las herramientas utilizadas como responsables del no éxito de los proyectos, por lo cual, es importante considerar para quienes se inician en la automatización de pruebas de software; el uso exitoso de una herramienta en una organización, no implica que este será de beneficio para otra organización. Depende de muchos otros factores que se deben considerar, como son: los procesos, la calidad de las pruebas, el conocimiento de las personas, la arquitectura y la abstracción con que se escriben los *test* de prueba. Este libro expone los factores a considerar para no ser parte de las estadísticas fallidas.

Es importante entonces considerar también estos otros factores, además de la herramienta Selenium, de esta forma probablemente se lograría alcanzar los objetivos propuestos en la presente tesis.

3. *TESTING* MANUAL EN LOS SISTEMAS TI

En este capítulo se describe en detalle el contexto funcional, tecnológico, capacidades de desarrollo y el procedimiento de *testing* manual de los sistemas en la compañía de TV que administra TI Chile.

3.1. Contexto de los sistemas

En la compañía de TV, existe un sistema principal y que por políticas de seguridad no podemos nombrar, y para efectos del presente documento de tesis será mencionado como sistema CORE. La función principal, del sistema CORE, es subir al satélite la parrilla de canales de los distintos operadores, tales como: FOX+, HBO, CDF, entre otros. Y también proveer las funcionalidades para la gestión de servicios al cliente, permitiendo por ejemplo: activar, cancelar, cambiar programación de las tarjetas de los decodificadores instalados en los clientes.

Existen otros sistemas en Chile que dan soporte a las áreas de negocio: Ventas, Telecenter, SelfService, BackOffice, Operaciones y Gestión. Estos sistemas se denominan sistemas satélites, definidos así, porque son un complemento al sistema CORE, permitiendo ampliar con nuevas funcionalidades no existentes. Estos sistemas fueron desarrollados por los equipos de TI Chile y TI Latinoamérica, quienes realizan modificaciones evolutivas y correctivas.

En total, existen 42 sistemas que permiten operar y soportar el modelo de negocio de la compañía de TV en Chile. En la imagen 8, se aprecia los más relevantes. Todos estos sistemas fueron construidos para ser utilizadas desde los navegadores web de las estaciones de trabajo, como estándar se utiliza Internet Explorer 9, versión recomendada por el sistema CORE para el correcto funcionamiento de este. La herramienta licenciada para el desarrollo de los sistemas satélites es Visual Studio 2010 y la base de datos (BD) que almacena la información es Oracle 11g.

En promedio se solicitan 20 requerimientos mensuales logrando ejecutar alrededor de 15 utilizando la capacidad máxima de 1.700 horas hombre (HH) de los equipos TI Chile (1.100 HH) y los equipos de Latinoamérica (600 HH).

Los sistemas más requeridos para implementar nuevas funcionalidades desde las áreas de negocio son: el sistema de ventas y en el sistema de atención de clientes, debido a las ofertas de promociones que se lanzan en el mes, provocando en algunos periodos del año un backlog de requerimientos debido a la mayor demanda respecto a las capacidades de horas disponibles para abordarlas. La capacidad máxima asignadas para desarrollo evolutivo y correctivo mensual es de 200 HH para el sistema de ventas y de 360 HH para el sistema de atención de clientes.

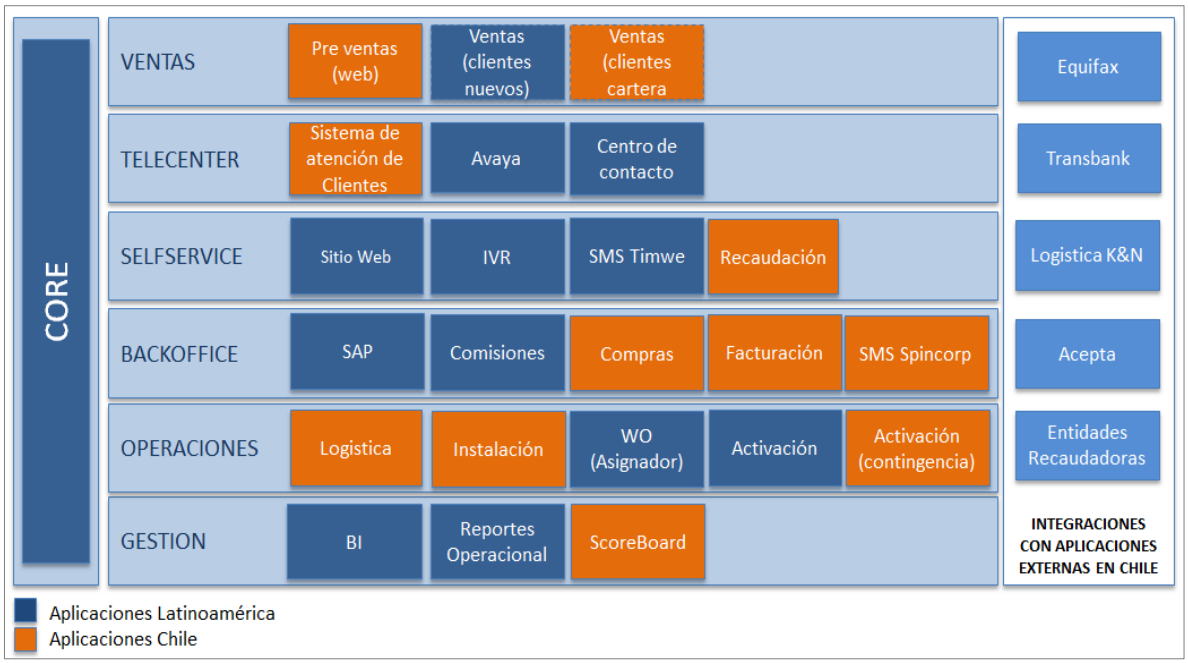

Imagen 8: Mapa de aplicaciones de la compañía TV Chile.

3.2. Proceso de desarrollo

En esta sección se muestra el proceso de desarrollo que guía la modificación de alguna pieza de software de las aplicaciones TI en Chile y Latinoamérica. Como se aprecia en la imagen 9, se trata de un proceso muy sencillo, donde al final de cada uno de los subprocesos se entrega un artefacto y/o documento que sirve como input al siguiente subproceso.

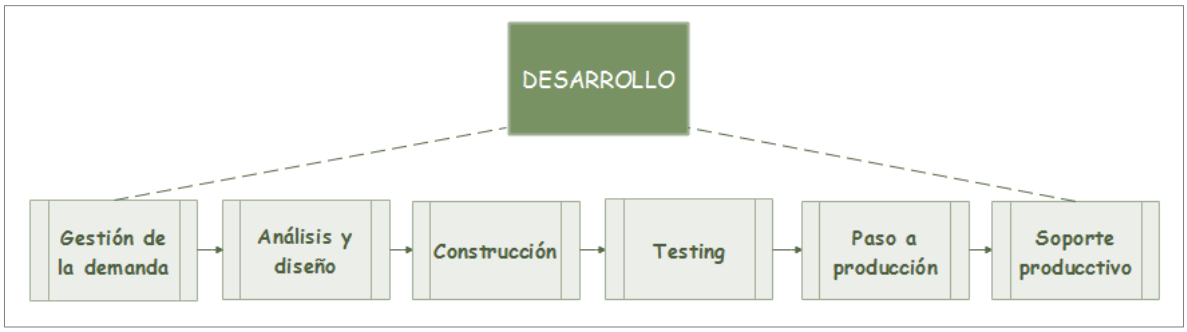

Imagen 9: Proceso de desarrollo de aplicaciones TI.

Se presenta a continuación el alcance de cada uno de los subprocesos, sin embargo, para el subproceso de *testing* se proporciona una descripción detallada, al ser el foco de mejora de este proyecto.

3.2.1. Subproceso Gestión de la demanda

Se realiza el análisis preliminar de la necesidad de negocio, con el objetivo de definir el alcance a desarrollar en los sistemas de TI. A partir del documento de alcance se realiza la evaluación de esfuerzo (HH) por parte de los equipos de SD, permitiendo realizar el análisis de costo / beneficio a presentar al comité formado por los directores y gerentes senior de la compañía TV Chile y así obtener la priorización de ejecución y fechas límites para aquellos requerimientos muy urgentes alineando la estrategia con la visión de la compañía TV. En la salida de este subproceso, el gestor de demanda entrega un plan de ejecución de alto nivel a los coordinadores de ejecución y el documento actualizado del alcance.

3.2.2. Subproceso Análisis y diseño

Tiene como objetivo el desarrollo del documento de especificación del requerimiento y un plan de ejecución en detalle. Se definen los ciclos de los entregables, las líneas base de las aplicaciones y las funcionalidades a desarrollar y probar en cada ciclo en función de las fechas críticas. Como salidas de este subproceso, el coordinador de SD entrega el plan de ejecución a los equipos de desarrollo y el documento de especificación del requerimiento.

3.2.3. Subproceso Construcción

Tiene como objetivo la modificación de las piezas de software estipuladas en el documento de especificación técnico. Como salida de este subproceso el equipo de desarrollo de SD entrega un nuevo producto que se agrega a una nueva línea base que se debe probar.

3.2.4. Subproceso *Testing*

Tiene como objetivo validar el producto, es decir, comprobar si cumple con las necesidades del cliente que están estipuladas en los documentos de alcance y el documento de especificación de requerimientos. Además, se realizan las pruebas de regresión al producto que permitan validar funcionalidades existentes previamente en producción.

Se estudian las nuevas funcionalidades del producto para definir el alcance de las pruebas en cada uno de los ciclos definidos en el plan. Se escriben los nuevos casos de prueba en función de los cambios.

Las pruebas para todos los sistemas son ejecutadas de forma manual y dependiendo del sistema a validar, se necesitarán al menos 2 semanas en la ejecución de las pruebas de validación para cada nuevo release.

En la imagen 10 se presenta en detalle el subproceso de *testing*.

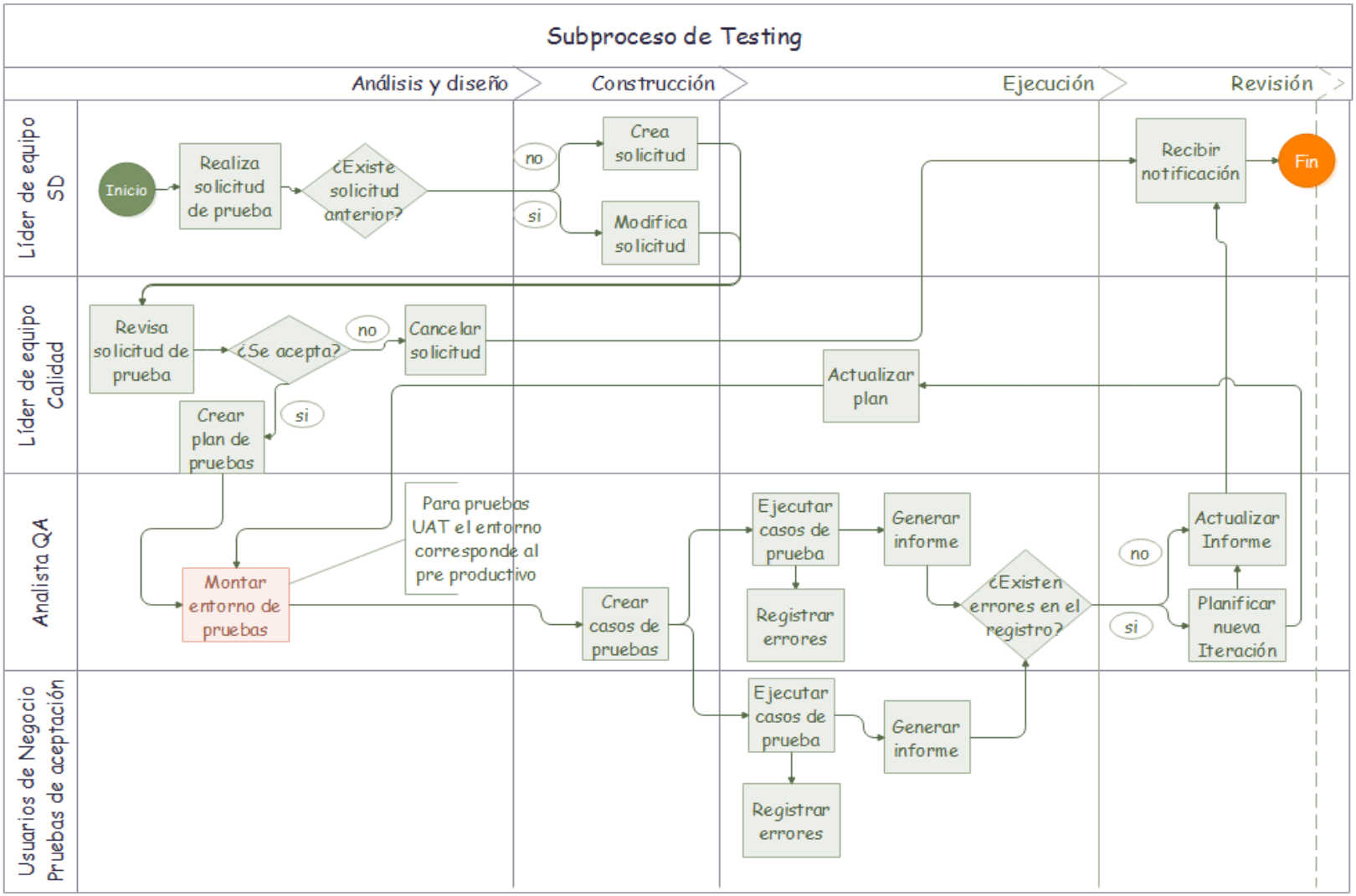

Imagen 10: Subproceso de *testing* existente.

En este subproceso, el líder del equipo de SD es quien realiza la solicitud de prueba enviando el documento de solicitud al líder del equipo de calidad. Este documento podría ser completado a partir de una solicitud anterior.

La solicitud debe indicar el alcance de la prueba, el contexto de los cambios realizados en el sistema, el plan de ejecución y el entorno donde se debe realizar. Para pruebas de aceptación de usuarios debe haber existido anteriormente la validación del equipo de calidad.

El líder del equipo de calidad es quien dirige el proceso de prueba. El analista QA es quien realiza el diseño de los casos de prueba a partir de las especificaciones del producto, gestiona el entorno de prueba (desarrollo o *testing*) y ejecuta los CP junto a los usuarios de negocio. Durante la ejecución se requiere documentar las pruebas en un informe. Al finalizar las pruebas de la iteración, este informe es revisado por el analista QA para suministrar y planificar con el líder de calidad una nueva iteración en el caso de existir errores y/o para entregar al líder de SD el resultado final con el informe de las evidencias.

Como contexto adicional, el sistema de ventas y en el de atención clientes, las pruebas de regresión no se alcanzaban a ejecutar totalmente en cada proceso de prueba, se abordaban generalmente solo las pruebas referidas al cambio, quedando un porcentaje importante de pruebas sin poder ejecutarse producto de compromisos adquiridos que obligan a su implementación en producción en las fechas límites solicitadas.

3.2.5. Subproceso Paso a producción

Tiene como objetivo implementar los cambios correctamente en ambiente de producción cumpliendo con la normativa de seguridad establecido, participan los equipos de SD quienes suministran los documentos del resultado de las pruebas de validación del proceso de *testing* y el documento del paso a paso para la realización del cambio y el equipo de operaciones infraestructura quienes realizan la instalación, validan la secuencia de pasos en la preparación del ambiente pre-producción.

Para formalizar un paso a producción se debe proporcionar el documento del checklist de aprobación del usuario funcional quien se apoya del informe de resultado de las pruebas para la toma de decisión.

3.2.6. Subproceso Soporte productivo

Tiene como objetivo asegurar el correcto funcionamiento del cambio una vez instalado en el ambiente de producción, generalmente se asigna un día completo en el monitoreo generando revisiones que permiten establecer evidencia de buen funcionamiento. El equipo participante de esta etapa son los responsables de

llevar a cabo el cambio (líder de SD, líder de calidad y algunos analistas participantes del desarrollo y prueba).

4. DESARROLLO DEL PRIMER PROTOTIPO

En este capítulo se describe en detalle la construcción de la herramienta de automatización para pruebas de regresión cuya implementación se hace a través de un prototipo en el sistema de ventas. Se presenta el alcance definido a partir del análisis realizado, se visualiza el diseño, la construcción y la etapa de transición en el subproceso de *testing*.

4.1. Alcance del prototipo

El alcance de la solución se presenta en la estructura de desglose de trabajo (EDT) elaborada para facilitar la comprensión del prototipo realizado.

Como se aprecia en la imagen 11, el prototipo de automatización de los casos de prueba, piloteado en el sistema de ventas de la compañía de TV, inicia con la etapa de análisis para la elaboración del levantamiento de requisitos funcionales (RF) y no funcionales (RNF) que fueron considerados para en la solución. También se realiza el levantamiento de los casos de prueba existentes, analizando la cobertura funcional de estos.

En la etapa de diseño se elabora una prueba de concepto y/o factibilidad de la solución inicial de acuerdo a los requisitos especificados, permitiendo desarrollar el diseño de arquitectura, el diagrama de secuencia de los objetos y el diagrama de componentes.

La etapa de construcción fue realizada en 3 iteraciones, en cada una de ellas se elaboran los *scripts* manuales y a partir de estos se realiza la trasformación en *scripts* automatizados.

Finalmente, en la etapa de transición se modifica el proceso de *testing* para incorporar las pruebas automatizadas del sistema de ventas.

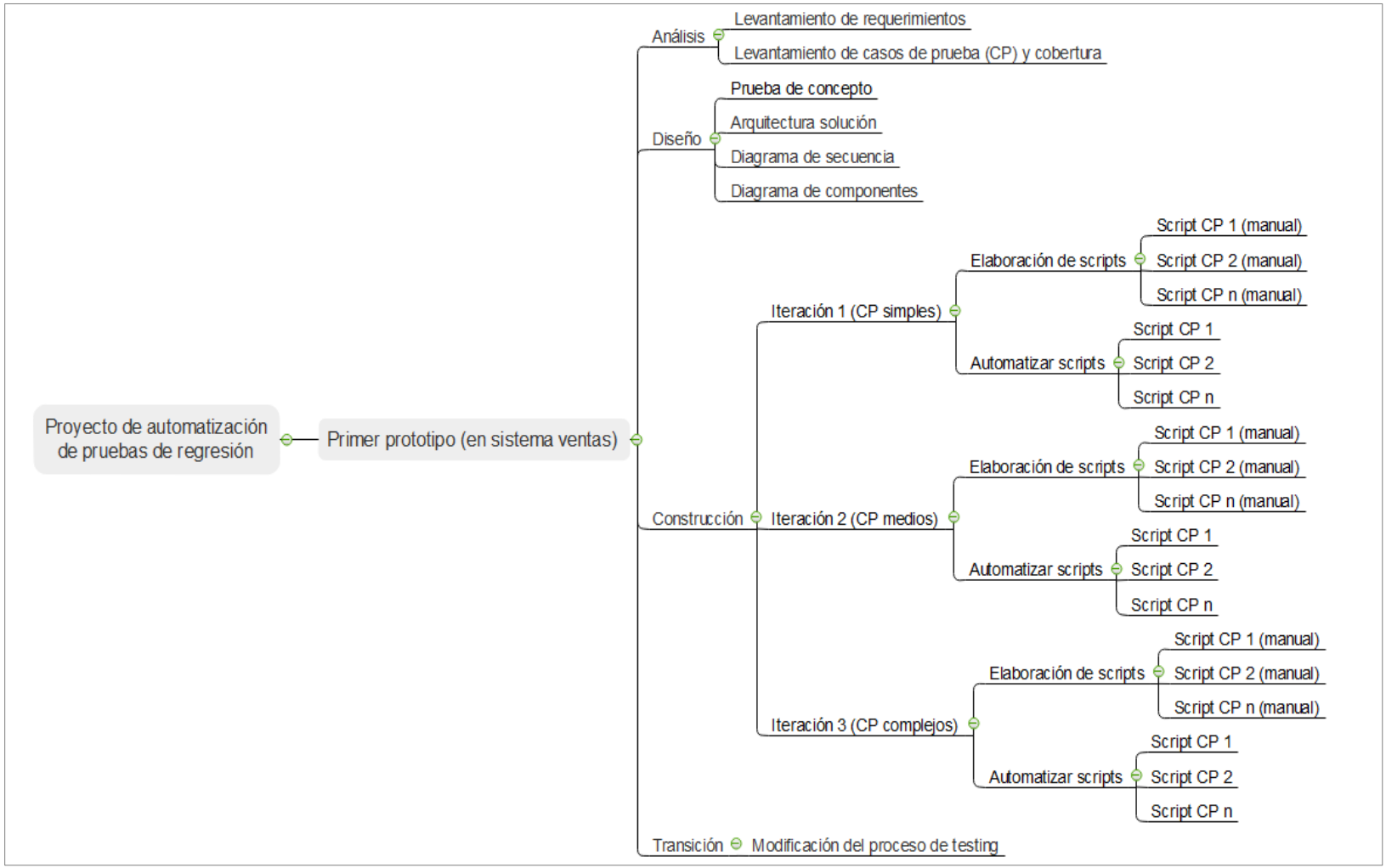

Imagen 11: EDT del prototipo.

4.2. Análisis

En esta sección se determina la funcionalidad del prototipo para realizar las pruebas de regresión de forma automática. Se presentan a continuación los objetivos específicos del prototipo, los requisitos funcionales y las restricciones identificadas como importantes a considerar.

4.2.1. Especificación de objetivos y requisitos del sistema de *testing*

En esta subsección se presentan los objetivos específicos, requisitos y restricciones que se deben considerar para el desarrollo del sistema de *testing* automatizado, elaborados a partir de entrevistas realizadas al gerente de SD y al equipo de calidad de SD.

Los objetivos específicos serán expresados como (OE), los requisitos serán expresados como requisitos funcionales (RF) y requisitos no funcionales (RNF), y las restricciones sobre tecnologías a usar serán expresadas como (RT).

Objetivos específicos:

OE1: Se debe revisar el nivel de cobertura actual de los casos de prueba existentes respecto de la funcionalidad del sistema elegido para el piloto, eliminando la redundancia de aquellos CP que pudiesen estar repetidos. El listado obtenido se considerará de acuerdo a la definición como casos de prueba del tipo regresión.

OE2: Se debe seleccionar los casos de prueba que serán automatizables, el criterio a considerar debe ser respecto al beneficio obtenido en comparación del esfuerzo a realizar. Por ejemplo, los casos de prueba que indican validar el formato de impresión de un reporte, donde el esfuerzo de automatizar demanda muchas horas de desarrollo y la validación a través del ojo humano es muy sencilla, serán clasificados con complejidad alta y no serán automatizables. Solo se debe automatizar tareas repetitivas y/o tediosas que consumen mucho tiempo manual y que amenazan la eficiencia. Un factor importante a no perder de vista son los objetivos del proyecto respecto del porcentaje de pruebas que se deben automatizar que permitiría obtener los beneficios perseguidos.

OE3: En la construcción de los *scripts* automatizados es muy importante considerar la facilidad en el mantenimiento posterior, esto implica obtener *scripts* en lenguajes de programación que dominen los desarrolladores del equipo SD, idealmente puedan ser mantenidos por el equipo de calidad de TI con conocimientos básicos de programación.

Requisitos funcionales:

RF1: Crear los casos de prueba de acuerdo a la matriz funcional del sistema de ventas. Considerar, por cada caso de prueba se debe realizar su correspondiente *script* de prueba.

RF2: Grabar los *scripts* de prueba (emular las acciones de un usuario final en la interacción con la aplicación piloteada) para transformarlos en *scripts* automatizados.

RF3: El sistema de *testing* debe permitir al QA seleccionar uno o más *scripts* automatizados, para probar funcionalidades específicas.

RF4: El sistema de *testing* debe permitir al QA programar la fecha y hora de inicio de la ejecución, permitiendo pruebas fuera del horario laboral del equipo de calidad de SD.

RF5: El sistema de *testing* debe permitir al QA modificar los datos de entrada de los *scripts* automatizados para la ejecución de las pruebas, sin tener que editar estos *scripts*.

RF6: El sistema de *testing* debe permitir al QA ejecutar un mismo *script* con varios registros de entrada de datos, donde cada registro representa una nueva prueba.

RF7: El sistema debe permitir al QA revisar los resultados de ejecución de los *scripts* ejecutados por intermedio de un reporte, mostrando los *scripts* de prueba ejecutados y en caso de haber un error, tiene que quedar registro del punto donde ocurrió dentro de las distintas validaciones del flujo de la prueba.

Requisitos no funcionales:

RNF1: Desde el punto de vista usabilidad

- El sistema debe ser fácil de usar, esto implica, que los nuevos usuarios puedan desarrollar una interacción efectiva con el sistema a través de sesiones de capacitación y la generación de conocimientos previos respecto a la materia. En una escala de 1 (poco usable) y 7 (muy usable), el sistema deberá ser valorado al menos con nota 4 que implica que el sistema es suficientemente fácil de ser usado.
- El sistema debe contar con manual de usuario al final del proyecto para entregar a los nuevos usuarios como materia de apoyo.
- El sistema debe proveer mensajes, ayuda contextual e información útil en línea cuando se está preparando una prueba y que permita orientar al usuario que lo utiliza.

RNF2: Desde el punto de vista organizacional

- Involucrar desde un inicio del proyecto al equipo QA en la concepción del proyecto para evitar resistencia al cambio en etapas de implementación, deben estar involucrarlos en todas las etapas del desarrollo del proyecto, en toda las decisiones a través de las distintas reuniones planificadas.

Restricciones sobre tecnología a usar.

RT1: La herramienta a utilizar para la automatización de los *scripts* debe:

- Ser gratuita, ya que no se desea realizar inversión de compra de licencias para los prototipos o utilizar la herramienta Visual Studio ya licenciada.
- Ser compatible con plataforma Microsoft.
- Ser compatible con el navegador Internet Explorer 9, acorde al estándar que define el sistema CORE.
- Permitir grabar para emular las acciones de un usuario final en la interacción con la aplicación piloteada.

RT2: Por motivos de seguridad, el sistema no podrá conectarse directamente con las bases de datos donde se almacena la información necesaria para ejecutar las pruebas. Esta información podrá ser leída desde una planilla Excel que se proporciona al equipo de calidad de la gerencia SD y la cual cumple con algunos criterios de seguridad, como por ejemplo, la información es generada contemplando ofuscamiento de los datos.

4.2.2. Análisis de cobertura de los casos de prueba

Esta subsección tiene como objetivo revisar el nivel de cobertura de los casos de prueba de regresión existentes. Se indica el método que permitió eliminar la redundancia de CP, identificar funcionalidades de prueba no cubiertas y se describe cómo se seleccionaron las pruebas que se iban a automatizar.

Para poder revisar el nivel de cobertura, se creó una matriz de trazabilidad funcional donde se listo las funcionalidades del sistema de ventas, permitiendo asociar los casos de prueba existentes con las funcionalidades de esta aplicación utilizada en el piloto.

Como antecedente, el equipo de calidad de SD a cargo de las pruebas exploratorias manuales contaba con listados de casos de prueba, sin embargo, no se tenía la relación con la funcionalidad de negocio. La construcción de la matriz de trazabilidad permitió visualizar redundancia de casos de prueba con algunas funcionalidades y problemas de cobertura por la inexistencia de casos de prueba para algunas funcionalidades del sistema.

Para redefinir los casos de prueba se utilizó la técnica de identificación a través de los casos de uso. En esta técnica se deben consideran todos los escenarios posibles.

Cada escenario posible corresponde a un caso de uso. Por ejemplo, para la función de negocio procesamiento de venta de un nuevo cliente, podríamos tener como escenarios posibles: crear preventa, llenar preventa, editar preventa y anular preventa. Para cada uno de estos escenarios corresponde crear un caso de uso a los que se debe identificar el flujo básico a seguir en el proceso y además, identificar los distintos flujos alternativos. En resumen, cada caso de uso o escenario se inicia con el flujo básico y después, combinando el flujo básico con el flujo alternativo se pueden identificar los escenarios restantes.

Continuando con el ejemplo, el flujo básico en la creación de una preventa, corresponde a completar con los datos personales del cliente: rut, nombre, dirección, teléfonos, mail, y en flujos alternativos seleccionar los distintos servicios asociados a productos comerciales que se quiere adquirir como son: grilla de canales básica, canales a la carta y programación premium.

FUNCION DE NEGOCIO		**1**	**2**	**3**
1.0	PROCESAMIENTO BEST FIT			
1.1	CONSULTAR RUT EN BEST FIT	PBF 11-1	PBF 11-2	PBF 11-3
2.0	PROCESAMIENTO DE VENTA			
2.1	CREAR PRE VENTA	CPV21		
2.1.1	LLENAR PRE VENTA	LLPV 211-1	LLPV 211-2	LLPV 211-3
2.1.2	EDITAR PRE VENTA	EPV212-1	EPV 212-2	EPV 212-3
2.2	AGREGAR CARRO PRE VENTA	ACPV22-1	ACPV22-2	
2.2.1	AGREGAR PRODUCTO BÁSICO	APB221-1	APB221-2	APB221-3
2.2.2	EDITAR PROD. BÁSICO PRE VENTA	EPB222-1	EPB222-2	EPB222-3
2.2.3	AGREGAR DECOS PRE VENTA	ADPV223-1	ADPV223-2	ADPV223-3
2.2.4	AGREGAR DECOS ADICIONALES	ADA224-1	ADA224-2	ADA224-3
2.2.5	AGREGAR PREMIUM FLEX PRE VENTA	APF225-1	APF225-2	APF225-3
2.2.6	EDITAR PREMIUM FLEX PRE VENTA	EPF226-1	EPF226-2	EPF226-3
2.2.7	CANCELAR PREMIUM FLEX PRE VENTA	APF226-1	APF226-2	APF226-3
2.2.8	AGREGAR PREMIUM ADIC. PRE VENTA	APA228-1	APA228-2	APA228-3
2.2.9	EDITAR PREMIUM ADIC. PRE VENTA	EPA229-1	EPA229-2	EPA229-3
2.2.10	CANCELAR PREMIUM ADIC. PRE VENTA	APA2210-1	APA2210-2	APA2210-3
2.2.11	ELIMINAR PREMIUM PRE VENTA	APA2211-**X**		

Tabla 4: Matriz de trazabilidad funcional del sistema de ventas (versión parcial).

La tabla 4 (versión parcial de casos de prueba), lista las funciones de negocio del sistema de ventas, con sus casos de prueba asociados. Los códigos asignados a los casos de prueba corresponden a las letras iniciales de la función definida, hasta aquí la matriz tiene un enfoque funcional de lo que el sistema debiese hacer, sin embargo, también debe considerarse lo que el sistema no debiese hacer, para esto último se agregan más casos de pruebas, los cuales se identifican agregando al final de los códigos la letra “X” como se visualiza en la última fila de la tabla 4. Por ejemplo, la función de negocio llamada “Eliminar premium de pre venta” contradice la normativa de seguridad respecto a eliminación de datos, es factible cancelar una suscripción y no eliminar un registro.

El análisis de cobertura, a través de la matriz de trazabilidad funcional, deslumbro la falta de cobertura de casos de prueba que no se venían ejecutando con las pruebas manuales. En el sistema de ventas existían tan solo 70 pruebas de regresión y se esperaba automatizar cerca de 50, sin embargo, la matriz resultante permitió identificar 169 casos de prueba.

El siguiente paso fue determinar del total de casos de prueba definidos, cuáles serán los que formarán parte de la automatización y los casos de prueba que no serán automatizables porque no es factible verificarlos o tiene un costo alto su comprobación automática. En los dos siguientes casos se optó por no automatizar:

- No se automatizarán pruebas de la función de negocio reportes ya que los casos de prueba asociados tienen como alcance validar (información completa visualizada en pantalla, proceso de imprimir y uso de formatos correctos) donde el esfuerzo de una automatización sería más costosas que las pruebas exploratorias manuales.
- No se automatizarán pruebas de la función de negocio procesamiento de pagos, cuyos procesos sistémicos requiere de ambientes QA con otras entidades hoy inexistentes para este tipo de ambiente; las pruebas exploratorias manuales se hacen realizando transacciones por $1 y a apuntando a las entidades formales en producción.

Los casos de prueba que no cumplen los criterios antes descritos, formaron parte de la automatización, a los cuales se le realizó un análisis que permitió estimar la complejidad que significaría realizar la automatización. Esta complejidad se obtuvo del análisis de cuatro variables representadas por las letras A, B, C, D.

Alcance de las variables para los criterios de selección:

A. Input de datos, representa el número de datos de ingreso requeridas para ejecutar el caso de prueba. Por ejemplo, la función de negocio “Consultar Rut en best fit” requiere que sea ingresado el Rut a consultar (un solo dato como input).

B. Validaciones, representa el número de validaciones que se deben realizar para poder ejecutar el caso de prueba, siguiendo el mismo ejemplo, el caso de prueba PBF 11-1 debe realizar 12 validaciones, una de ellas es validar que el Rut corresponda a una persona mayor de 18 años (dato proporcionado por el Registro Civil integrado al sistema).
C. Pre-requisitos, representa al número de funcionalidades de negocio que deben haberse realizado antes de poder ejecutar este caso de prueba. Siguiendo con el mismo ejemplo, para poder llegar a consultar un Rut de un cliente se tiene que haber realizado correctamente la autenticación del usuario donde se valida que tenga el perfil adecuado para este tipo de consultas.
D. Pasos de ejecución, corresponde al total de acciones que se deben realizar para la ejecución de un caso de prueba. Siguiendo el mismo ejemplo, se debe ingresar este Rut en el formulario y presionar el botón buscar, esto corresponde a un total de dos acciones.

El valor de estas variables pudiera ser cero como valor mínimo si no tiene incidencia en el contexto que representa.

A cada una de estas variables se asignó un mismo peso (0.25) y se determinó la complejidad de automatizar bajo el siguiente criterio:

Baja: ponderación de complejidad menor 10.

Media: ponderación de complejidad mayor o igual a 10 y menor que 15.

Alta: ponderación de complejidad mayor o igual a 15.

Los valores definidos como extremos (10 y 15) fueron tomados en base a la experiencia manifestada por el equipo QA al revisar el listado de casos de prueba y agrupar según ponderación resultante.

Por ejemplo, la ponderación para la función de negocio “Consultar Rut en best fit” correspondiente a la fila 1.1 de la tabla 5, resulto de baja complejidad de acuerdo al siguiente cálculo de la fórmula:

Complejidad = (A*0.25) + (B*0.25) + (C*0.25) + (D*0.25)

= (1*0.25) + (12*0.25) + (2*0.25) + (6*0.25)

= 5 (equivale a baja complejidad)

Este método permitió agrupar los casos de prueba por el nivel de complejidad que representa según el resultado de evaluación de cada uno de ellos y de esta forma poder priorizar la ejecución del proyecto para iniciar el piloto de automatización. Es decir, se realizó la automatización comenzando por aquellos casos de prueba de

baja complejidad, siguiendo con los de complejidad media y finalizando con los de alta complejidad.

La tabla 5 muestra el resultado del análisis bajo los criterios antes indicados.

FUNCION DE NEGOCIO		Caso de prueba	A	B	C	D	Ponderación	Complejidad
1.0	PROCESAMIENTO BEST FIT							
1.1	CONSULTAR RUT EN BEST FIT	PBF 11-1	1	12	2	6	5	**baja**
2.0	PROCESAMIENTO DE VENTA							
2.1	CREAR PRE VENTA	CPV21	14	1	2	25	11	**media**
2.1.1	LLENAR PRE VENTA	LLPV 211-1	15	14	3	34	17	**alta**
2.1.2	EDITAR PRE VENTA	EPV 212-1	3	14	2	6	6	**baja**
2.2	AGREGAR CARRO PRE VENTA	ACPV22-1	15	14	4	35	17	**alta**

Tabla 5: Ponderación de complejidad de los casos de prueba (versión parcial).

4.3. Diseño

Se presentan a continuación el diseño de los *scripts* manuales, los diagramas conceptuales y sus respectivas arquitecturas físicas y lógicas que fueron generadas para la solución.

El primer diseño estuvo orientado a las pruebas de factibilidad técnica haciendo uso de la herramienta Selenium IDE en la automatización de las pruebas. Estas pruebas tenían como objetivo validar si esta herramienta era suficiente para cubrir los objetivos específicos, los requisitos y las restricciones para el prototipo.

El segundo diseño, con el enfoque centrado en los datos, permitió abordar limitaciones importantes encontradas en la validación de factibilidad. Una de ellas referido al momento de la ejecución de una nueva prueba, era necesario editar los *scripts* automatizados para modificar los datos de la prueba. Para este diseño se considera el uso de otras herramientas en la automatización, Visual Studio 2010 y Selenium WebDriver.

4.3.1. Diseño de *script* manual

De acuerdo a lo indicado en el marco teórico, los *scripts* de test manual son confeccionados a partir de los casos de prueba. Estos *scripts* deberán ser suficientemente claros para permitir la transformación en un *script* automatizado. A continuación, en la tabla 6, se presenta un ejemplo de cómo se deben especificar estos *scripts* de pruebas.

Objetivo del caso de prueba:	Consultar con el Rut del nuevo cliente si cumple las reglas definidas por administración venta. Por ejemplo, si el cliente tiene deuda pendiente con la compañía de TV.
Identificador:	PBF 11-1.
Nombre del caso de prueba:	Consultar Rut en *best fit.*
Precondiciones:	No aplica.
Pasos	Resultado esperado
1) Ingresar la url de la página de validación de clientes.	Se debe mostrar la interfaz de autentificación de usuario.
2) Ingresar usuario perfil vendedor en el input habilitado “User”.	
3) Ingresar password del vendedor en el input habilitado “Pwd”.	
4) Presionar el botón “Aceptar”.	Se debe habilitar la interfaz de validación de clientes.
5) Ingresar el Rut del nuevo cliente en el input habilitado “Rut cliente”.	
6) Presionar el botón “consultar”.	Se debe habilitar en la interfaz de ingreso de venta.

Tabla 6: Ejemplo de *script* manual de caso de prueba.

4.3.2. Diagrama conceptual de prueba de factibilidad utilizando Selenium IDE

En la imagen 12 se presenta el diagrama conceptual de la funcionalidad utilizando la herramienta Selenium IDE para grabar los *scripts* manuales (sección amarilla en la imagen 12) y reproducir las pruebas (sección celeste en la imagen 12) en el entorno real en el cual se ejecutarán las pruebas.

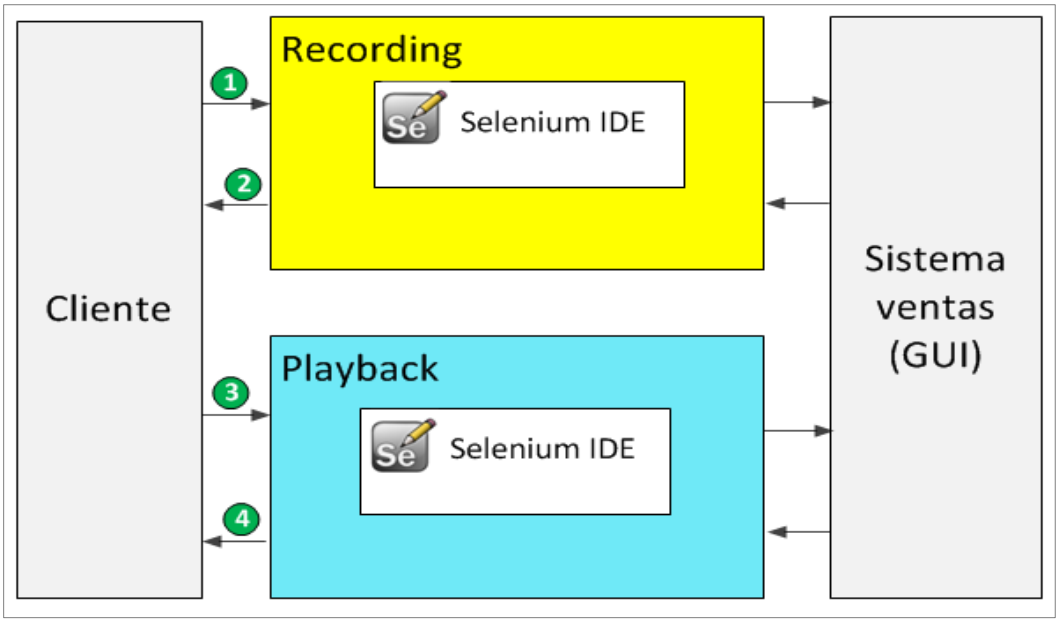

Imagen 12: Diagrama conceptual de automatización.

En el diagrama conceptual de la imagen 12, se visualizan 4 pasos que el cliente representado por el usuario QA debe ejecutar para realizar la automatización de un *script* de prueba, estos son:

1. El usuario QA graba los *scripts* de prueba obtenidos del proceso de análisis de pruebas existentes. Para esto, se ejecutan todos los pasos escritos en los casos de prueba. Por ejemplo, los pasos de la tabla 6. La herramienta Selenium IDE captura los eventos realizados en el sistema de ventas correspondientes a cada uno de estos pasos ejecutados.
2. El usuario QA al finalizar la ejecución del *script* puede seleccionar el lenguaje generado (C#, Java, Perl, PHP, Python y Ruby) con que se guardará el *script* automatizado.
3. En una etapa posterior, el usuario QA o cualquier otro que realice la función de *testing,* selecciona desde la interfaz de Selenium IDE, el *script* automatizado y ejecuta el *playback*, lo cual reproduce cada uno de los pasos previamente grabados en el *script* sobre el sistema de venta, proporcionando los mismos datos ingresados, botones del formulario presionados y la transición de tiempos según fueron capturados.
4. El usuario QA puede rescatar de la herramienta Selenium IDE el resultado de la ejecución para comparar con el resultado descrito en el *script* manual.

4.3.3. Resultado de la prueba de factibilidad utilizando Selenium IDE

Considerando los objetivos específicos, requisitos y restricciones definidos para este prototipo y al evaluar el resultado del proceso de grabado y reproducción de

los *scripts*. Selenium IDE permitió grabar un conjunto de *scripts* de prueba, sin embargo, al reproducir estos mismos *script*s, cuya lógica de capturas de grabación existían elementos de AJAX en la GUI del sistema de ventas, no se ejecutaron correctamente. Un mismo *script* ejecutado arrojaba resultados diferentes en condiciones de ejecución similar (servidor, datos de prueba, release el sistema de ventas, entre otros), en algunos casos se lograba completar el 100% y en otros quedaba inconcluso o directamente arrojaba error por lo que generó desconfianza respecto en la funcionalidad de la herramienta.

Se detectó también, complejidad para el mantenimiento de los *scripts* automatizados. Al editar estos y observar el código generado en JAVA o C#, no eran sencillos de modificar para los miembros del equipo de calidad de TI con poca experiencia en programación, requerido por el objetivo específico 3, para efectos de mantenibilidad, muy necesario para poder actualizar los casos de prueba grabados cuando las GUI cambian por nuevas funcionalidades.

De acuerdo a lo planteado en requisito funcional 5, no se pudo alcanzar en la automatización de los *scripts*, un procedimiento que permita modificar los datos de entrada de estos, teniéndose que editarlos e intervenir el código generado para modificar los datos de prueba, lo cual atacaba directamente el beneficio esperado de la automatización.

Considerando los problemas antes mencionados, fue necesario cambiar la arquitectura del prototipo y la herramienta de automatización, para rectificar y lograr los beneficios de la automatización. La nueva arquitectura se presenta en la siguiente subsección.

4.3.4. Diagrama conceptual de la solución

El diagrama conceptual de la imagen 13 representa el nuevo diseño conceptual que aborda el enfoque centrada en los datos. Se consideró en el nuevo diseño utilizar el IDE de Visual Studio 2010 para programar los *scripts* en conjunto con las API que expone Selenium WebDriver para el proceso de grabado y reproducción.

En el enfoque centrado en los datos, los *scripts* de prueba deben ser creados para que durante la ejecución de estos *scripts,* por intermedio de los identificadores *(Key)* puedan extraer desde una fuente externa alguna acción o los datos que serán utilizados para la ejecución de la prueba (ver Sección 2.4). Se puede apreciar en la imagen 13, la conexión del motor de ejecución (sección celeste de la imagen 13) con un archivo DataPool que contiene los datos que leerá el *script* automatizado al momento de la ejecución.

Las principales diferencias a favor del modelo conceptual de la imagen 13 en comparación con el mostrado en la prueba de factibilidad de la imagen 12, esta permite: separar la capa de presentación de los datos, controlar los tiempos

durante la ejecución de los *scripts* y aprovechar las ventajas que expone Selenium WebDriver, este último permite embeber el navegador IE para la realización de las pruebas. La principal desventaja de este diseño es que el IDE de Visual Studio no realiza la función de *recording*, como lo hace Selenium IDE, por lo que se debe codificar en Java o C# cada uno de los *scripts* por un programador con conocimientos en este tipo de solución.

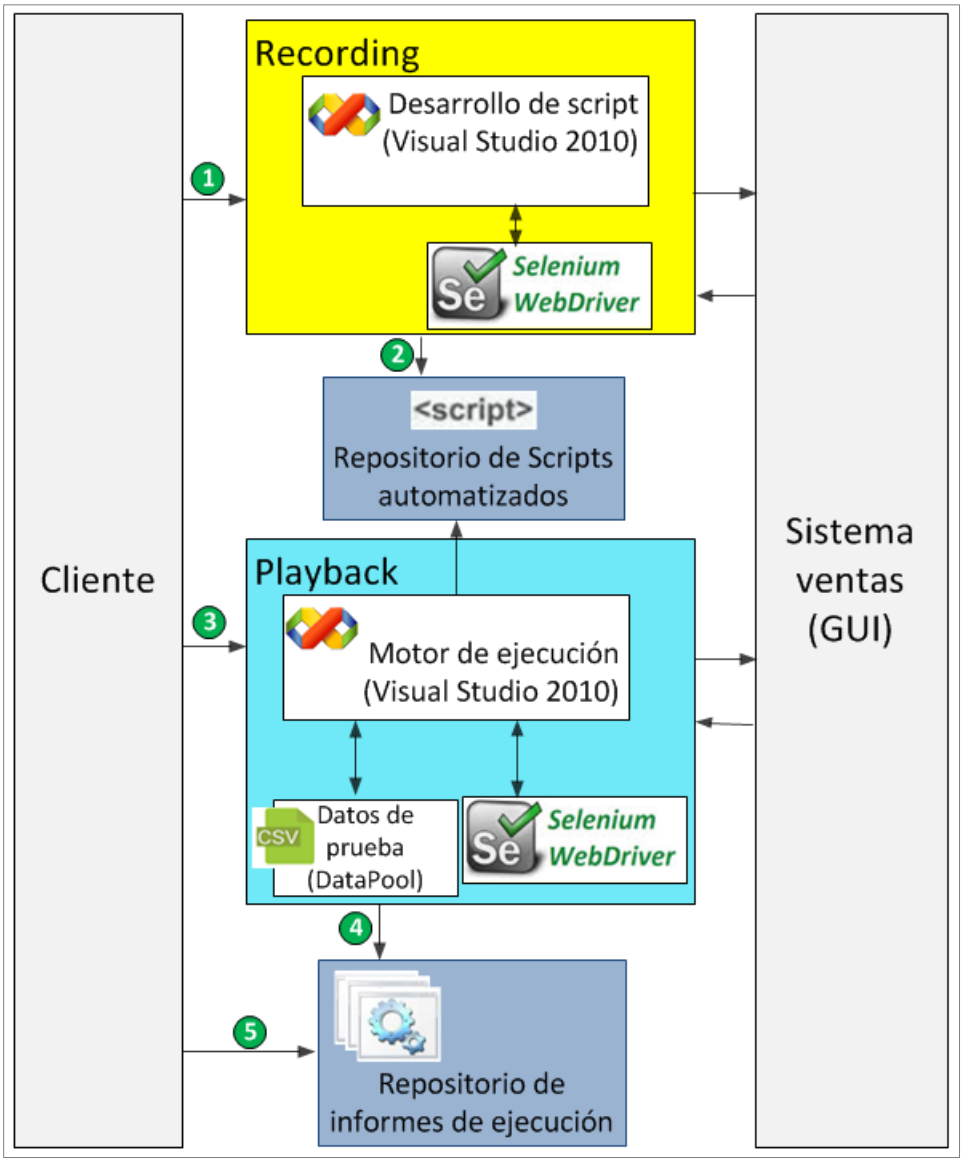

Imagen 13: Diagrama conceptual de la solución de automatización.

El usuario QA, seguirá definiendo los casos de prueba y tendrá también que preparar los datos del archivo DataPool, pudiendo ingresar para un mismo identificador de *script* varios registros con datos diferentes, de esta forma se

contempla, por ejemplo, realizar pruebas de validaciones con valores límite, útil en nuestro caso para revisar criterios de negocio.

El proceso de *testing* para un usuario QA inicia desde el punto 3 de la imagen 13, una vez que los *scripts* ya fueron codificados por el desarrollador, pudiendo:

- Acceder al motor de ejecución, que contiene los *scripts* ya automatizados y seleccionar uno o más de estos scripts para ejecutarlos.
- Acceder al repositorio de los reportes de ejecución para revisar el resultado de las pruebas realizadas. El contenido de cada uno de estos archivos describe el paso a paso del *script* ejecutado donde se puede identificar el caso de prueba que representa, los datos utilizados como input, y el punto preciso donde pudiera existir un error con su respectiva imagen de captura de la GUI, esto último, no lo provee Selenium IDE en los reportes de ejecución.

4.3.5. Arquitectura física de la solución

En la arquitectura de la imagen 14. Selenium WebDriver provee un conjunto de drivers los que permiten automatizar las pruebas directamente con cada navegador (Chrome, Firefox, Internet Explorer, entre otros), es decir, el drivers es instalado en la maquina cliente desde donde se ejecutarán los *scripts*, y en el servidor de aplicación donde se aloja la solución de automatización, lo cual permitirá embeber el navegador para la interacción con la interfaz del sistema de ventas y proveer todas las funcionalidades de IE.

La arquitectura soporta múltiples usuarios ya que usa mayoritariamente los recursos de la maquina cliente, permitiendo ejecutarse en varias instancias del navegador IE. El servidor de aplicación provee las funcionalidades que permiten ejecutar los *scripts* automatizados.

Imagen 14: Diagrama de arquitectura física.

4.3.6. Arquitectura lógica de la solución

Al proveer Selenium WebDriver los drivers necesarios que permiten hacer llamadas directas al navegador usando el soporte nativo de estos, se desarrolló una solución de aplicación web, considerando una arquitectura multicapas. Entre las ventajas de utilizar esta arquitectura, está la de poder aislar la lógica de la aplicación en componentes para lograr un sistema escalable, extensible y modificable, permitiendo que sus partes sean más reutilizables. En la imagen 15 se muestra la arquitectura lógica de la solución.

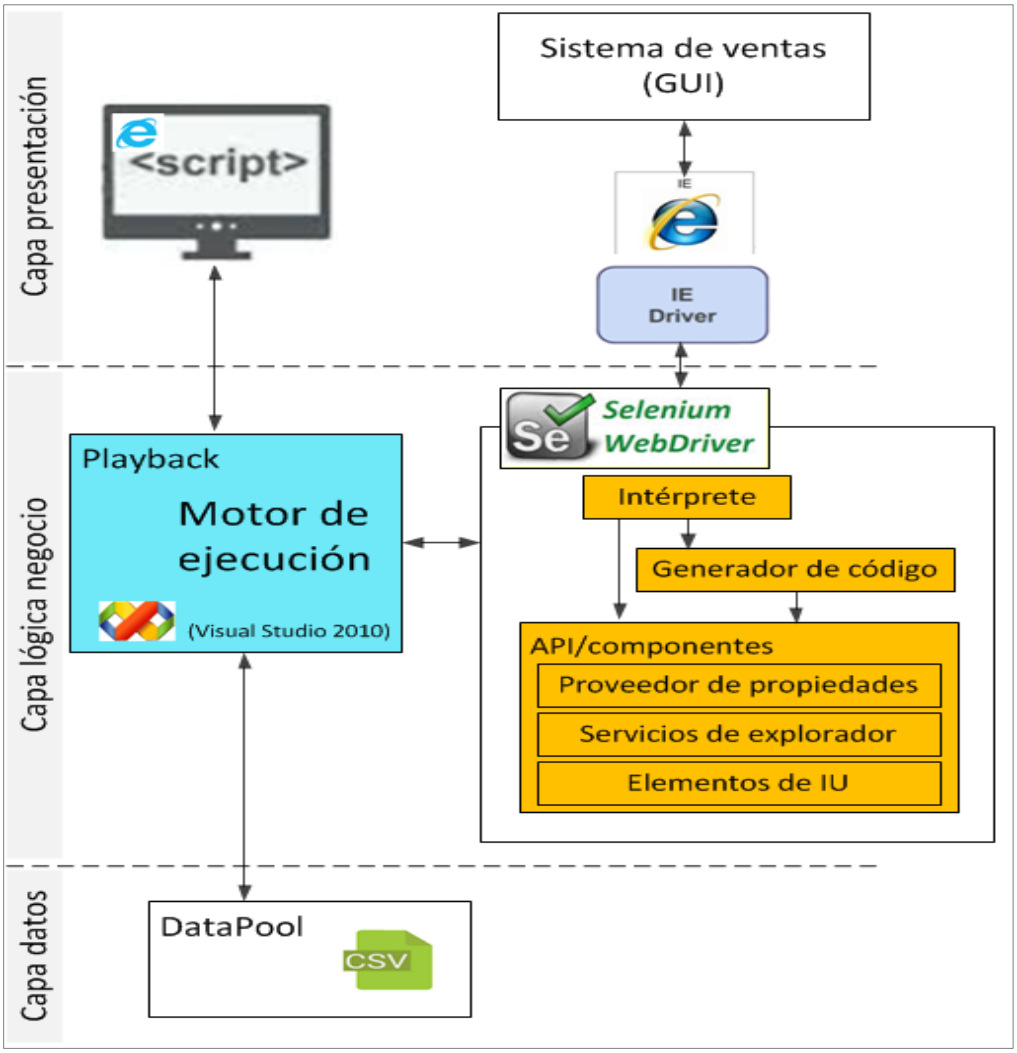

Imagen 15: Diagrama de arquitectura lógico.

La arquitectura de la imagen 15, tiene un enfoque centrado en los datos, la cual contempla en la capa de datos un archivo CSV llamado DataPool. Este archivo

contiene los datos de prueba que serán utilizados para la reproducción de las pruebas.

Siguiendo con el ejemplo de la Sección 4.3.1, después de mostrar cómo se deben especificar los *script* de prueba, se construye el archivo DataPool considerando las condiciones, campos y acciones especificadas de este *script*. La tabla 7 representa el *script* del caso de prueba y la tabla 8, los registros representan los datos requeridos por el *script* del ejemplo, los que fueron introducidos en la forma manual a este archivo.

El motor de ejecución visualizado en la imagen 15, fue programado para ejecutar las lógicas, secuencias requeridas y lectura de datos desde el archivo DataPool para cada uno de los casos de prueba. Es decir, en el ejemplo, el motor fue programado para ejecutar el caso de prueba “PBF 11-1”, siguiendo los pasos definidos de la tabla 7 y validando los resultados según se definen, para lo cual, extrae desde el archivo mostrado en la tabla 8, la Url, User, Pwd, pulsando el botón Aceptar, y validando que deriva a la Url de la fila 2, después de pulsar el botón Aceptar (funcionalidad correcta). Luego extrae, para el paso 4, el Rut del cliente, validando que no debe derivar a la Url de la fila 3 (funcionalidad incorrecta, el Rut de cliente no cumple reglas de validación, registra deuda pendiente con la compañía de Tv).

El beneficio principal del enfoque centrado en los datos es poder ejecutar varias pruebas (en el ejemplo de la tabla 8 contiene 3 pruebas), sin tener que editar los *scripts* automatizados, permitiendo separar lo lógica de negocio de los datos, situación que no fue factible en las pruebas de factibilidad de la Sección 4.3.2.

CASO PRUEBA	Objetivo CP	Identificador	Nombre CP	Precondiciones	Pasos	Resultado esperado
	Consultar si el Rut del nuevo cliente cumple las reglas definidas por administración venta. Por ejemplo, si el cliente tiene deuda pendiente con la compañía de TV.	PBF 11-1.	Consultar Rut en best fit.	False	1) Ingresar la url de la página de validación de clientes.	Se debe mostrar la interfaz de autentificación de usuario.
					2) Ingresar usuario perfil vendedor en el input habilitado “User”.	
					3) Ingresar password del vendedor en el input habilitado “Pwd”.	
					4) Presionar el botón “Aceptar”.	Se debe habilitar la interfaz de validación de clientes.
					5) Ingresar el Rut del nuevo cliente en el input habilitado “Rut cliente”.	
					6) Presionar el botón “consultar”.	Se debe habilitar en la interfaz de ingreso de venta.

Tabla 7: *Script* de prueba.

DATOS	N° Prueba	Identificador	Paso	URL	User	Pwd	Botón	Resultado	Rut Cliente	Nombres	Apellidos	Región	Comuna	Dirección
	1	PBF 11-1.	1	http://login.sistemas.cl	13459826	xxxxxx	Aceptar	true						
		PBF 11-1.	4	http://clientes.sistemas.cl			Aceptar	true	10638945-3					
		PBF 11-1.		http://ventas.sistemas.cl			Aceptar	*false*						
	2	PBF 11-1.	1	http://login.sistemas.cl	13459826	xxxxxx	Aceptar	true						
		PBF 11-1.	4	http://clientes.sistemas.cl			Aceptar	true	9996748-1					
		PBF 11-1.		http://ventas.sistemas.cl			Aceptar	true		Jose Tomas	Silva	Metropolitana	Nuñoa	Los alerces 2015
	3	PBF 11-1.	1	http://login.sistemas.cl	13459826	xxxxxx	Aceptar	true						
		PBF 11-1.	4	http://clientes.sistemas.cl			Aceptar	true	15761368-7					
		PBF 11-1.		http://ventas.sistemas.cl			Aceptar	true		Nicolas Alonso	Matínez	Araucania	Temuco	Los apostoles 01928

Tabla 8: Estructura del archivo DataPool (versión parcial).

4.4. Diagrama de componentes

En la imagen 16, se presentan los componentes: CORE y Selenium WebDriver, que forman parte de la solución construida en el prototipo. La componente Sistema de ventas, se muestra solo como referencia ya que los elementos fueron especialmente construidos para interactuar con las funcionalidades de esta aplicación.

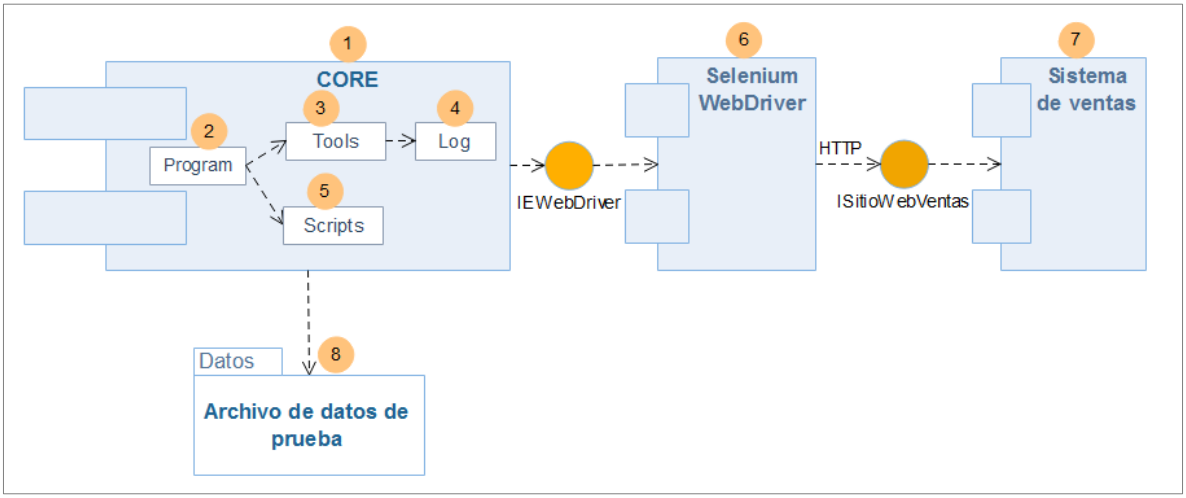

Imagen 16: Diagrama de los componentes para el prototipo.

En la tabla 9, se describen estos componentes y sus respectivas clases que conforman el diagrama de componentes. Además del paquete Datos que referencia a los archivos con los datos de prueba.

N°	Descripción
1	La componente CORE es la más relevante del Motor de ejecución (sistema de ejecución de pruebas automatizado), proporciona y consume el comportamiento a través de interfaces. En esta componente se desarrollaron las clases que permiten la automatización de los casos de prueba.

Tabla 9: Descripción de elementos y componentes

2	La clase Program dirige la ejecución de las pruebas automatizadas e inicializa a las otras clases para la ejecución de los *scripts*. Es decir, permite dirigir, invocando los métodos y funciones de las clases para la

	reproducción de un *test* de ejecución.
3	La clase Tools contiene todas las funciones y métodos utilizadas para la interactividad entre la aplicación a probar y el navegador IE, en otras palabras, es la encargada de buscar, validar, asignar valores o cualquier tipo de interacción sobre los objetos o elementos de los formularios contenidos en el sistema de ventas.
4	La clase Log permite escribir el paso a paso de la reproducción de un *script* automatizado. Esta clase contiene todas las funciones y métodos que utiliza la clase Tools para registrar el resultado al finalizar una tarea.
5	La clase *Scripts* contiene la lógica de ejecución de cada caso de prueba creado en la automatización de pruebas del sistema de ventas.
6	La componente Selenium WebDriver es uno de los 4 componentes de la suite de Selenium descrito en la Sección 2.5. La componente permite interactuar con el navegador como si se tratase de acciones ejecutadas por usuarios directamente sobre el browser y de forma independiente a la aplicación que se está probando.
7	La componente Sistema de ventas, se muestra como referencia al sistema que se utilizó para la ejecución de las pruebas.
8	El paquete Datos, representa al archivo que contiene la información para cada uno de los *scripts* automatizados que forman parte de la ejecución de los casos de prueba. Un ejemplo del contenido de uno de estos archivos se visualizó en la tabla 8.

Tabla 9: Descripción de elementos y componentes

4.5. Descripción de colaboración entre los objetos

En la imagen 17, se presenta la secuencia de comunicación entre los principales componentes al momento de realizar la reproducción de los *scripts* automatizados en las pruebas de regresión.

El usuario QA, interactúa con la componente CORE para seleccionar los *scripts* que se van a reproducir. Esta componente reproduce la ejecución de todos los *scripts* seleccionados y a través de la componente Selenium WebDriver permite la interacción con la GUI del sistema de ventas.

La componente CORE ejecuta en cada *script* la secuencia de los pasos que fueron programados de acuerdo al caso de prueba. Estos pasos están representados por las acciones del *script*, y en cada acción ejecutada se escribe el resultado en el reporte final. Este reporte permite revisar la completitud de ejecución con los posibles errores para análisis del mismo.

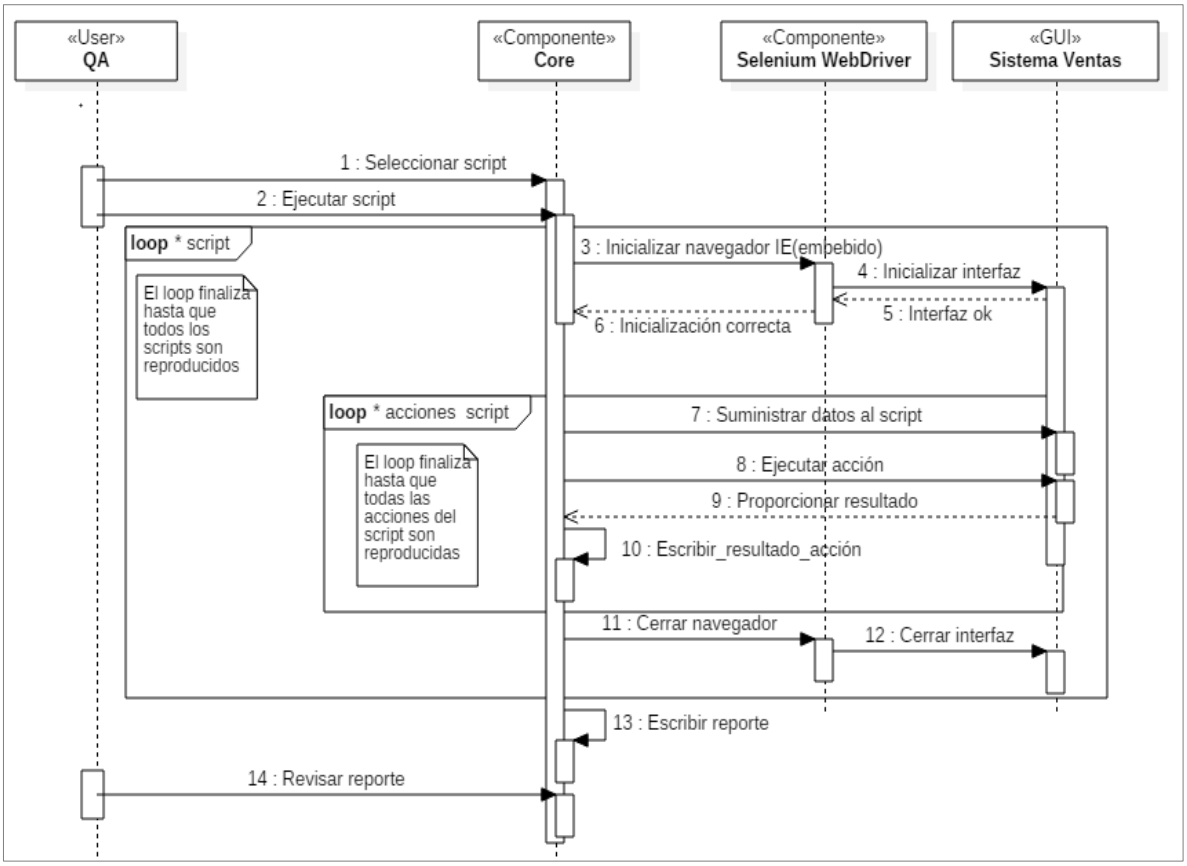

Imagen 17: Diagrama de secuencias para prototipo 1.

4.6. Construcción

En esta sección se presenta una breve explicación de la metodología seleccionada para el desarrollo del prototipo.

4.6.1. Metodología utilizada

La metodología iterativa incremental fue la más idónea para abordar la construcción del prototipo. Por cada nueva iteración que se realizó correspondió un incremento en la solución de automatización, cuyo entregable permitió desde ese momento disponer de los *scripts* automatizados para realizar pruebas de regresión.

Se realizaron 3 iteraciones para la construcción del prototipo. En cada una de las iteraciones se ejecutaron las actividades: diseño manual de los *scripts*, ejecución manual de los *scripts*, codificación de los *scripts*, pruebas de ejecución de los *scripts* automatizados. Para el diseño manual de los *scripts* se escribió el paso a paso de la secuencia de prueba, tal como se mostró en el ejemplo de la Sección 4.3.1. En la ejecución manual, se valida la secuencia escrita para comparar el

resultado esperado y el resultado obtenido. Una vez efectuada la validación de todos los CP contemplados para la iteración, se procede a la codificación de los *scripts,* transformándolos en *scripts* automatizados*.* La iteración finaliza al reproducir automáticamente los casos de prueba, validando los resultados al igual que la ejecución manual antes realizada.

Es Importante mencionar que las iteraciones fueron ejecutadas de acuerdo al grado de complejidad que representan los casos de prueba, partiendo la construcción por aquellos de menor complejidad. Esta recomendación es recogida a partir de los distintos casos de estudios citados por [Graham, 2012], donde se plantea como una buena estrategia es iniciar con funcionalidades de automatización simples para luego incorporar otras de mayor dificultad.

4.7. Transición

En esta sección de transición se presenta la modificación temporal en el proceso de desarrollo (temporal mientras finaliza el proyecto de automatización), específicamente se modificó el subproceso de *testing* presentada anteriormente en la Sección 3.2.4.

En la imagen 18, se muestra el mismo proceso que en la imagen 8 de la Sección 3.2.7, donde ahora las actividades nuevas se presentan marcadas de color amarillo. Como se puede ver en esta imagen, se realizaron pocas alteraciones con el objetivo de realizar una transición al nuevo proceso de forma simple, para no generar errores en la ejecución del nuevo proceso.

A continuación, se explican los principales cambios. El líder del equipo de calidad TI, cuando acepta una solicitud de una prueba (SP), debe revisar si el sistema a probar tiene CP con *scripts* automatizados, de tener CP estos son revisados junto al analista QA desde la matriz funcional existente para seleccionar los que se tendrán que ejecutar automáticamente como pruebas de regresión y planificar considerando además, el escenario de ejecución manual para las nuevas funcionalidades.

El analista QA, con el plan de pruebas que se entrega obtiene el entorno donde se ejecutarán las pruebas (desarrollo, *testing*, UAT). Para los CP a ejecutar de forma automatizada, se debe preparar los datos de prueba que tendrán que leer al momento de la ejecución. Finalmente el último cambio fue al proceso de registrar errores ya que para las pruebas automatizadas deber realizar la interpretación del informe de resultados que se crea en cada ejecución de los *scripts* de *test* automatizados.

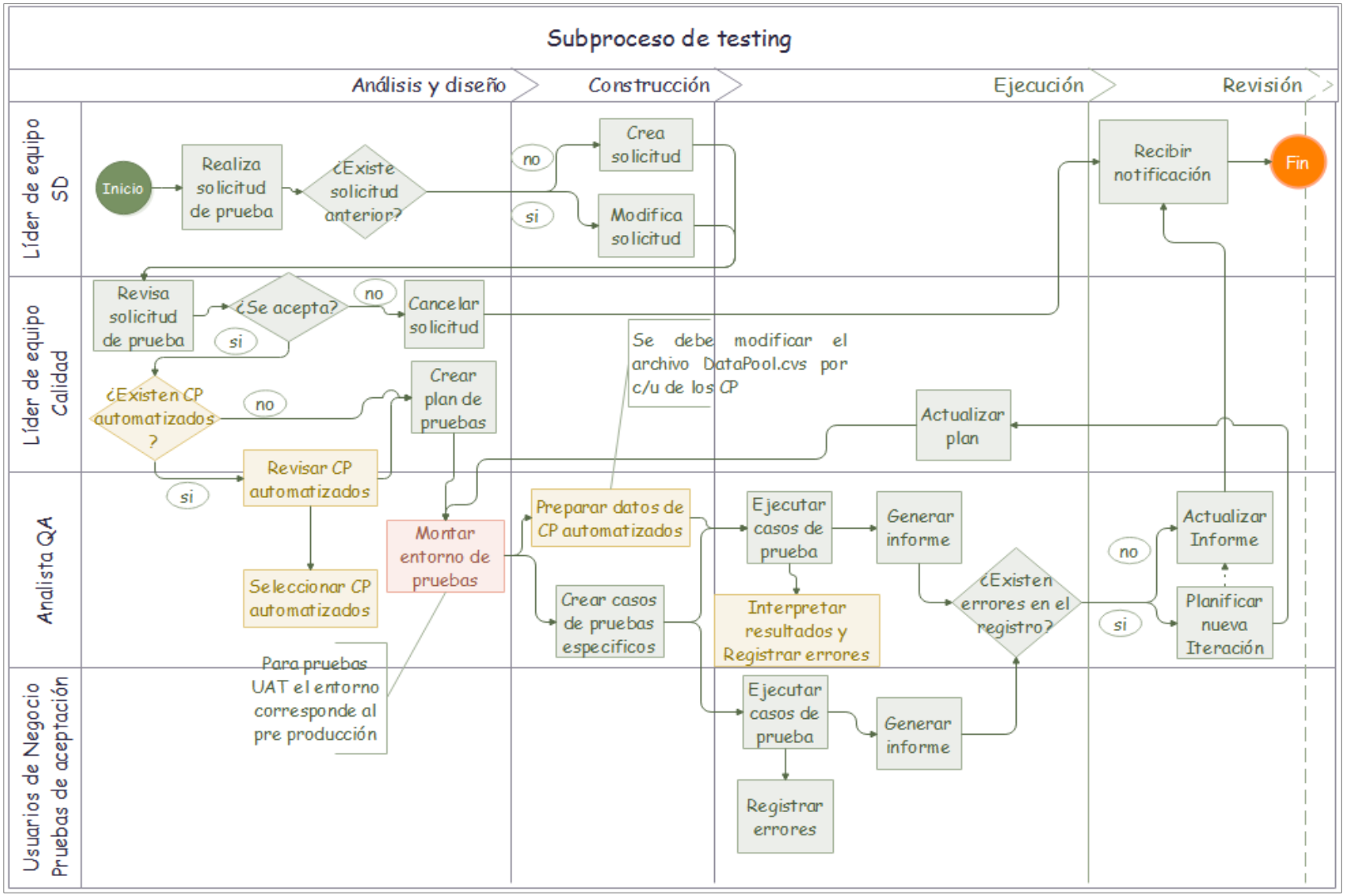

Imagen 18: Nuevo proceso de *testing* para etapa de transición.

4.8. Lecciones aprendidas

En esta sección se identifican algunas lecciones aprendidas de carácter preliminar que fueron surgiendo durante el desarrollo del prototipo.

4.8.1. Lecciones positivas

1. Cuando no existe conocimiento técnico de una herramienta por ningún integrante de un equipo de proyecto, las pruebas de factibilidad son buena alternativa para introducir conocimiento. La prueba realizada a la herramienta Selenium IDE permitió corregir de forma temprana el método, el diseño y la herramienta pensada para abordar el proyecto de automatización producto de los resultados obtenidos. Permitió también detectar que la solución inicial propuesta tenía un alto costo de mantenibilidad asociado (por cada nueva ejecución se debe editar los *scripts* grabados para modificar los datos de entrada con los que se ejecutarán las pruebas).
2. En forma preliminar, se detectó que la estrategia de iniciar la codificación de los *scripts* por aquellos CP más simples, usando una herramienta conocida (Visual Studio, apoyado de la componente de Selenium WebDriver), permitió internalizar el conocimiento necesario para desarrollar los CP de mayor dificultad y afrontar el proyecto de una forma más optimista al definir objetivos realistas y alcanzables.
3. La metodología de desarrollo iterativa incremental permitió:
    - Realizar los ajustes necesarios al inicio de cada nueva iteración y no esperar hasta el final como en otras metodologías, disminuyendo el riesgo del desconocimiento existente en esta materia de automatización de pruebas.
    - Obtener *scripts* automatizados de forma más rápida. Los *scripts* automatizados de la primera iteración se entregaron a menos de 2 meses iniciado la construcción quedando disponibles para utilización del equipo de calidad productiva.

4.8.2. Lecciones negativas

1. Hubo resistencia al cambio de las personas que participaron en el proceso de prueba, principalmente debido a la creencia de que serán reemplazados en su rol por una herramienta.
2. Hubieron errores involuntarios del procedimiento manual para preparar los datos de prueba y asociar en el formato predefinido de cada CP. Muchas ejecuciones fallidas fueron producto de la información asociada no era la correcta para el CP, impidiendo la ejecución completa.
3. El tiempo involucrado para programar un *script* automatizado fue de 6 HH para un CP simple, de 12 HH para un CP de complejidad media y de 16 HH para un CP de complejidad alta. Este tiempo es excesivo pensando en querer replicar este modelo para el resto de los sistemas satélites de TI en Chile.

4. Los cambios en el sistema de ventas afectan considerablemente a los *scripts* automatizados, con lo cual se tenía que volver a codificar los *scripts* impactados, destinando al menos la mitad del tiempo del esfuerzo original planificado en la construcción inicial.

Este primer prototipo fue pensado para proporcionar agilidad al proceso de *testing* del sistema de ventas y principalmente para detectar posibles errores a pruebas de regresión que no se venían haciendo por falta de recursos (tiempo y personas) cada vez que se realizaba un nuevo release de este sistema. Esta solución permitió entregar una herramienta que ayudo al objetivo planteado, sin embargo, también permitió, plantearse nuevos desafíos en el ámbito de automatización de pruebas que permitiera abordar las mejoras respecto a las lecciones negativas y resaltar aquellas lecciones consideradas como positivas.

Al analizar más profundamente el ámbito y las características de la herramienta construida, se considera que podría aportar beneficios a otros sistemas de la compañía si se realiza los ajustes que permitan proporcionar la agilidad al proceso de *testing*.

5. DESARROLLO DEL SEGUNDO PROTOTIPO

En este capítulo se describe en detalle la construcción de dos nuevos módulos de la herramienta de automatización para pruebas de regresión cuya implementación se hace a través de un segundo prototipo en el sistema de atención de clientes. En la construcción de la herramienta, estos nuevos módulos permitirán grabar las pruebas (sin codificación) y podrán asociar datos a la pruebas. Se presenta el alcance definido, se visualiza el diseño, la construcción y la etapa de implementación en el subproceso de soporte productivo.

5.1. Alcance del prototipo

El alcance de la solución se presenta en la estructura de desglose de trabajo (EDT) elaborada para facilitar la comprensión del segundo prototipo realizado.

Como se aprecia en la imagen 19, el nuevo prototipo de automatización inicia con la etapa de análisis para la elaboración del levantamiento de requisitos funcionales (RF) y no funcionales (RNF) que fueron considerados para la solución. También se realizó el análisis de cobertura funcional de los casos de prueba existentes para el sistema de atención de clientes.

En la etapa de diseño, se realizó el diagrama conceptual, arquitectura física y lógica, diagramas de componentes y de secuencia de los objetos.

En la etapa de construcción, se utilizó la metodología cascada permitiendo desarrollar previamente los dos nuevos módulos indicados y modificar el motor de ejecución de *script*, para poder iniciar la automatización de los scripts. Los módulos referidos son: grabador de *scripts,* generador de datos de prueba, y el motor de ejecución. Este último desarrollado a partir de la componente CORE del primer prototipo y adaptado en la nueva arquitectura. Como parte de la construcción se realizó también la elaboración de los *scripts* manuales que representan los CP, y finalizando con la transformación en *scripts* automatizados.

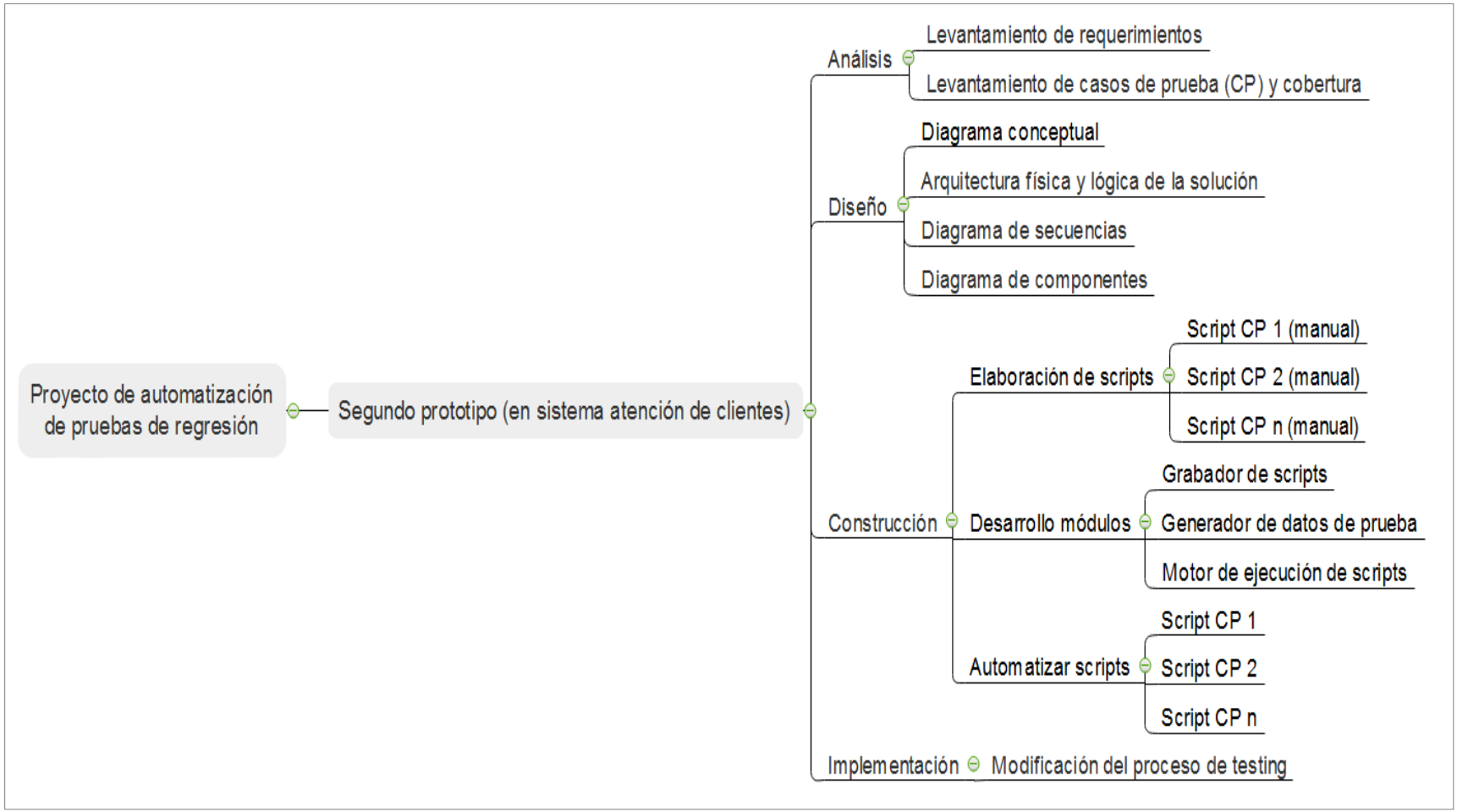

Imagen 19: EDT del segundo prototipo.

5.2. Análisis

En esta sección se determina la funcionalidad deseada del segundo prototipo para realizar las pruebas de regresión de forma automática en el sistema de atención de clientes.

5.2.1. Especificación de requisitos del sistema de testeo

Para el desarrollo de la solución, fueron considerados todos los objetivos específicos (OE), restricciones técnicas (RT), requisitos funcionales (RF) y requisitos no funcionales (RNF) definidos para el primer prototipo en la Sección 4.1.1, por lo cual, no se vuelven a escribir y solo se añaden en esta subsección los nuevos OE y RF a considerar.

Los nuevos OE que se plantearon para este prototipo abordan mayoritariamente los problemas de las lecciones aprendidas de la Sección 4.8.2.

OE4: Es importante mantener actualizados los *scripts* en la medida que se modifican las funcionalidades en el sistema de atención de clientes para que sean incluidos en forma oportuna en la automatización de pruebas, por lo cual, el tiempo y esfuerzo que requiere este procedimiento es muy importante para solucionar el problema detectado en el primer prototipo.

OE5: Se debe disminuir el tiempo que lleva desarrollar la automatización de un caso de prueba y la preparación de los datos de prueba existente en del primer prototipo.

Requisitos funcionales:

RF8: Crear un módulo sistémico que permita grabar el paso a paso descrito en los *scripts* de prueba y transformarlos en *scripts* automatizados, sin la necesidad de codificación.

RF9: Crear un módulo sistémico que permita seleccionar y asociar los datos de prueba a los *script* automatizados, eliminando la escritura de estos en forma manual como se hizo en el primer prototipo.

RF10: Crear un módulo sistémico que permita seleccionar los test de ejecución de acuerdo a las funcionalidades que se requieren validar y que además, permita planificar la fecha y hora de la reproducción automática de los test.

5.2.2. Análisis de cobertura de los casos de prueba

Esta subsección tiene como objetivo revisar el nivel de cobertura de los casos de prueba de regresión existentes.

Al igual que el primer prototipo, para poder revisar el nivel de cobertura, se creó una matriz de trazabilidad funcional, en la cual se listan las funcionalidades del sistema de atención de clientes, asociando los respectivos CP existentes de esta aplicación.

Esta matriz permitió visualizar la existencia de redundancia de CP en algunas funcionalidades de la aplicación y también la inexistencia de cobertura para algunas otras funcionalidades del sistema.

Para redefinir los CP se utilizó la misma técnica de identificación por medio de casos de uso indicada en la Sección 4.2.2 del primer prototipo.

La tabla 10 (versión parcial), lista las funciones de negocio que un cliente puede requerir a la compañía de TV, las que representan las funcionalidades del sistema de atención de clientes. Por ejemplo, la solicitud de un nuevo equipo, la migración del plan contratado por un cliente son funcionalidades que el sistema soporta. Las columnas del 1 al 3, representa los CP a realizar y Los códigos asignados a los casos de prueba corresponden a las letras iniciales de la función definida.

Como resultado se obtuvieron 296 CP, que representan las pruebas de regresión que se deben realizar cada vez que el sistema de atención de clientes es modificado. **FUNCION DE NEGOCIO**		**1**	**2**	**3**
1.0	EQUIPOS ADICIONALES			
1.1	Cliente solicita nuevos equipos SD	SE11-1	SE11-2	SE11-3
1.2	Cliente solicita nuevos equipos HD	SE12-1	SE12-2	SE12-3
1.3	Cliente solicita nuevos equipos DVR HD	SE13-1	SE13-2	SE13-3
2.0	CAMBIO DE PLAN			
2.1	Clientes solicita cambio plan (Upgrade)	UP21-1	UP21-2	UP21-3
2.2	Clientes solicita cambio plan (Downgrade)	DW22-1	DW22-2	DW22-3
3.0	CAMBIO DE PLAN & ADQUIERE EQUIPO			
3.1	Clientes solicita cambio plan (Upgrade) & nuevo equipo SD	UP&SE31-1	UP&SE31-2	UP&SE31-3
3.2	Clientes solicita cambio plan (Upgrade) & nuevo equipo HD	UP&SE32-1	UP&SE32-2	UP&SE32-3
3.3	Clientes solicita cambio plan (Upgrade) & nuevo equipo DVR HD	UP&SE33-1	UP&SE33-2	UP&SE33-3

Tabla 10: Matriz de trazabilidad funcional del sistema de atención de clientes (versión parcial).

El siguiente paso fue determinar del total de casos de prueba definidos, cuáles serán los que formarán parte de la automatización y los casos de prueba que no serán automatizables porque no es factible verificarlos o tiene un costo alto su comprobación automática. En los dos siguientes casos se optó por no automatizar: verificación de reportes impresos y la validación de procesamiento de pagos, por inexistencias de ambientes de *testing* por parte de las entidades bancarias.

Del total señalado, es factible automatizar un total de 200 CP, los que fueron seleccionados aplicando el criterio antes descrito.

A diferencia del primer prototipo, no fue necesario realizar un análisis de complejidad de los CP seleccionado. La solución de automatización permitirá la captura de las secuencias sin que sea necesaria la codificación de cada uno de estos.

5.3. Diseño

Se presentan a continuación el diagrama conceptual, la arquitectura física y lógica que fueron generadas para la solución.

El diseño de los *scripts* es idéntico al presentado en la Sección 4.3.1, por lo cual no se vuelve a escribir en esta sección.

5.3.1. Diagrama conceptual

En la imagen 20, se presenta el diagrama conceptual de la funcionalidad utilizando los tres módulos construidos. En la cual se puede apreciar al módulo grabador de *scripts* para capturar las secuencias de los *scripts* manuales, al módulo generador de datos de prueba para asignar los datos a los *scripts* automatizados y por último, el módulo motor de ejecución, para reproducir los test de ejecución. Dos de los tres módulos descritos utilizarán la componente de Selenium WebDriver para embeber el navegador web IE que permitirá la interacción con la interfaz de la aplicación de atención de clientes.

El procedimiento para poder grabar, asignar datos y reproducir las pruebas es el que se describe a través de la numeración mostrada en la imagen 20. El cliente estará representado por el usuario QA. Este usuario podrá:

1. Grabar los *scripts* de prueba obtenidos del proceso de análisis de pruebas existentes. Para esto, se ejecutan todos los pasos escritos en los casos de prueba. Por ejemplo, los pasos de la tabla 11. La herramienta de automatización a través del módulo Grabador de *script* captura los eventos realizados en el sistema de atención de clientes correspondientes a cada uno de los pasos ejecutados.

2. Extraer los archivos que fueron generados como *script* automatizados para cargar en el paso 3.
3. Asignar datos de prueba a los *script*, para lo cual, deberá cargar dos archivos a la herramienta de automatización a través del módulo Generador de datos de prueba, estos son: El *script* de prueba que se desea ejecutar y el archivo que contiene los datos de prueba para asociarlos al *script*.
4. Extraer los archivos que fueron generados como *test* de ejecución para cargar en el paso 5.
5. Planificar la ejecución de los *test*, para lo cual, tendrá que cargar los test seleccionados e ingresar la fecha y hora de ejecución que se desea realizar la reproducción de esto. Esta funcionalidad es útil para dejar programado su ejecución fuera de horario laboral, por ejemplo: en un proceso nocturno.
6. Podrá revisar posteriormente el resultado de la ejecución a través de los reportes que fueron generados.

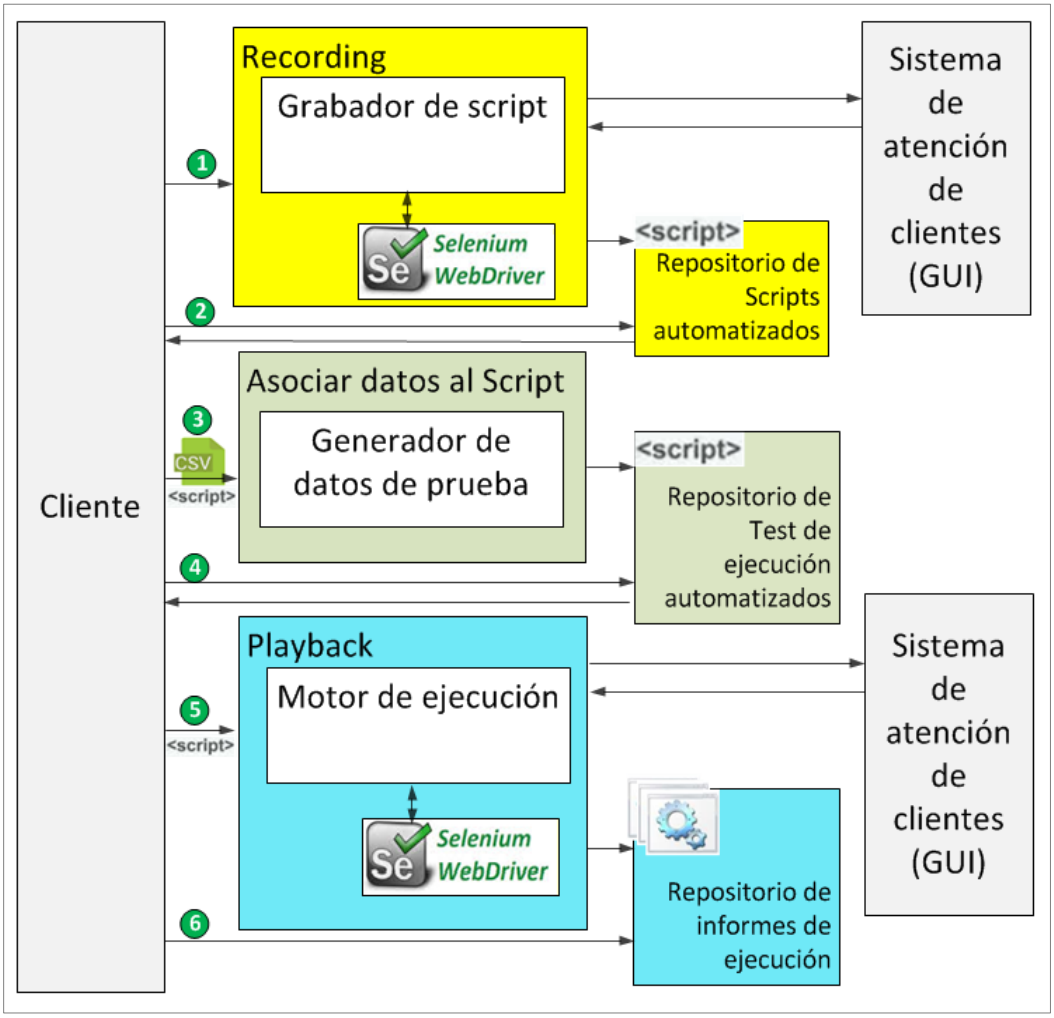

Imagen 20: Arquitectura automatización con enfoque centrada en los datos.

CASO PRUEBA	Objetivo CP	Identificador	Nombre CP	Precondiciones	Pasos	Resultado esperado
	Cliente solicita adquirir un quipo estándar (SD) adicional a los que tiene	SE11-1	Cliente solicita nuevos equipos SD	False	1) Ingresar la url del sistema de atención clientes.	Se debe mostrar la interfaz de autentificación de usuario.
					2) Ingresar usuario perfil vendedor en el input habilitado “User”.	
					3) Ingresar password del vendedor en el input habilitado “Pwd”.	
					4) Presionar el botón “Aceptar”.	Se debe habilitar la interfaz del sistema de atención de clientes.
					5) Ingresar el Rut del cliente en el input habilitado “Rut cliente”.	
					6) Presionar el botón “consultar”.	Se debe habilitar en la interfaz la información del cliente (Plan contratado, equipos existentes en su suscripción y disponibilidad (máximo 6 unidades por cliente).
					7) Agregar nuevo equipo	Se debe mostrar en la interfaz el costo de instalación ($).
					8) Presionar el botón “Aceptar”.	Se debe crear orden de trabajo y se debe habilitar en la interfaz la fecha de instalación para selección.
					9) Ingresar fecha de instalación en el input habilitado “Fecha instalación”.	
					10) Presionar el botón “Aceptar”.	Se debe modificar la orden de trabajo con la fecha ingresada en la interfaz.

Tabla 11: Ejemplo de *script* de prueba.

5.3.2. Arquitectura física de la solución

En la imagen 21, se presenta la arquitectura física de la solución, donde el cliente está representado por browser IE, desde la cual se invoca a la solución de automatización publicado en un servidor de aplicación Web IIS, en el cual se instala los módulos de la herramienta de automatización y Selenium WebDriver desde donde se ejecutaran los *scripts* ya preparados, los drivers de Selenium permitirán embeber el navegador para la interacción con la interfaz del sistema de atención de clientes y proveer todas las funcionalidades de IE.

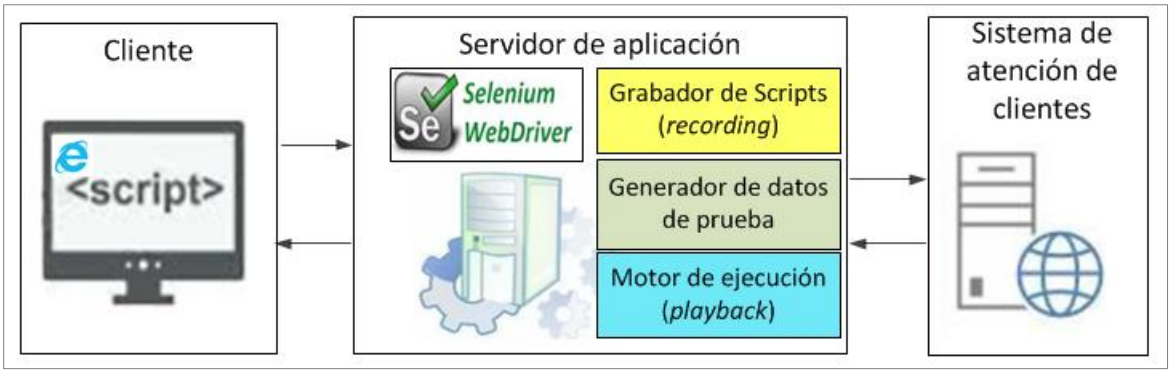

Imagen 21: Diagrama de arquitectura física.

5.3.3. Arquitectura lógica de la solución

En la imagen 22, se presenta la arquitectura lógica de la solución en tres niveles. La capa de presentación corresponde a GUI de los tres módulos anteriormente descritos, donde solo dos de ellos interactúan con la GUI del sistema de atención de clientes a través de Selenium WebDriver. La capa de negocio está compuesta por las componente CORE, la componente Selenium WebDriver. Por último la capa de datos, compuesto por un archivo DataPool.

Esta arquitectura, desde el punto de vista automatización de pruebas sigue teniendo un enfoque centrado en los datos y la única diferencia del archivo DataPool respecto de la visualizada en el ejemplo de la Sección 4.2.8, en esta versión del diseño no se necesita indicar por cada registro el identificador del caso de prueba que representa, solo debe contener los datos de prueba que serán utilizados para la reproducción de estas. En la tabla 12 se muestra una versión parcial de los campos del archivo que será utilizado por el módulo generador de datos de prueba.

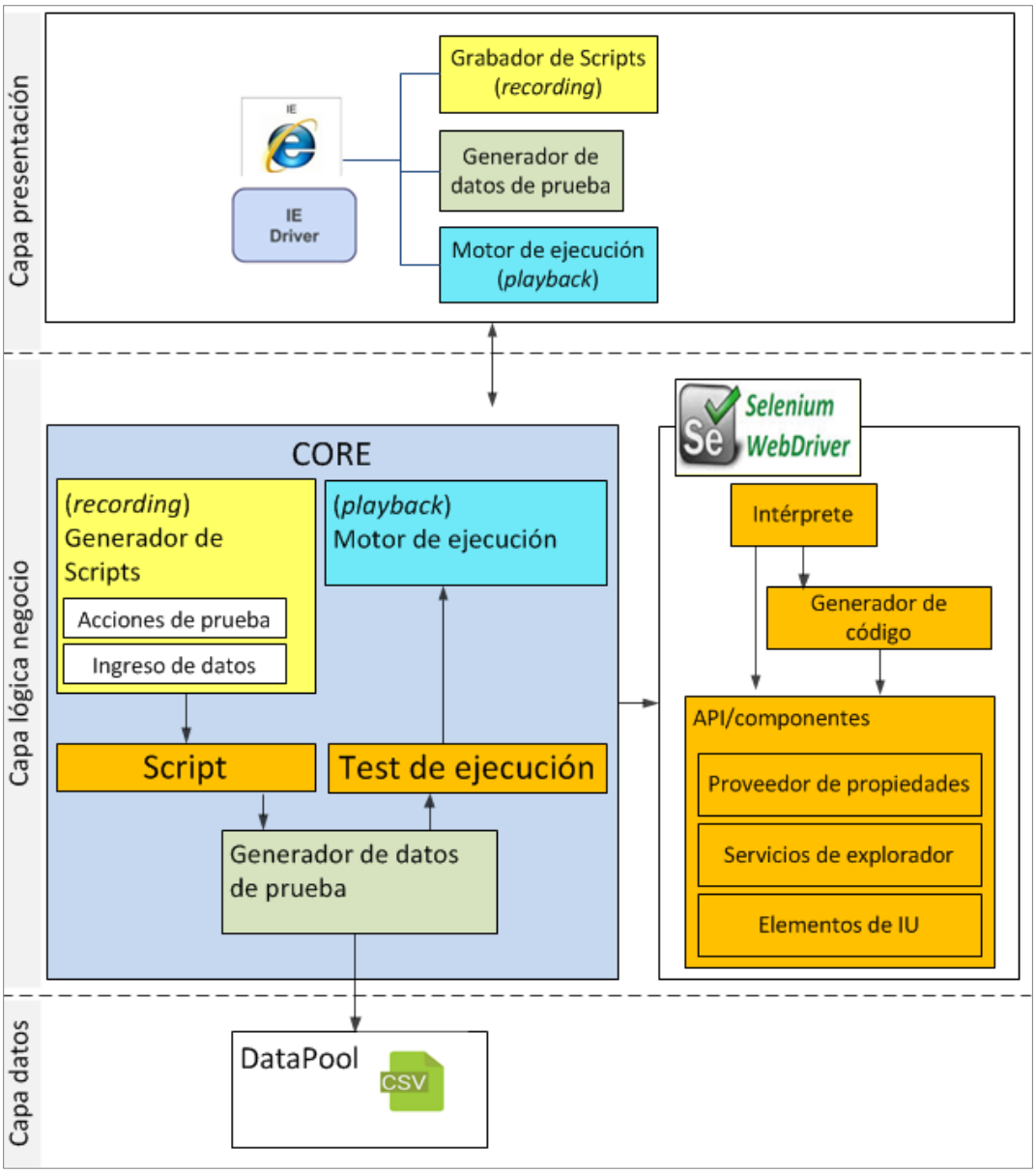

Imagen 22: Diagrama de arquitectura lógico.

USER	CLAVE	RUT_CLIENTE	NOMBRES	PRIMER APELLIDO	SEGUNDO APELLIDO
13459826-3	xxxxxxxx	4096561-1	JUAN ANDRES	SOTO	SOTO
13459826-3	xxxxxxxx	2021234-2	PEDRO	ZUÑIGA	SALES

Tabla 12: Estructura del archivo DataPool (versión parcial).

5.4. Diagrama de componentes

En esta sección, se presentan los componentes de la solución. En la imagen 23, se muestran los componentes: CORE y Selenium WebDriver, que forman parte de la herramienta construida en el nuevo prototipo. La componente Sistema de atención de clientes, se muestra solo como referencia ya que las clases que fueron construidos en las subcomponentes de CORE permiten interactuar con los elementos de esta aplicación.

En la tabla 14, se describen estos componentes y sus respectivas clases que conforman el diagrama de componentes. Además de los paquetes que referencian a los archivos generados. No se volverá a especificar la componente Selenium WebDriver, está ya fue descrita en la Sección 4.4.

N°	Descripción
1	La componente CORE es la más relevante de la nueva herramienta de automatización, proporciona y consume el comportamiento a través de interfaces. En esta componente se desarrollaron 3 subcomponentes, estos son: *Recording* perteneciente al módulo Grabador de *script*, Datos de prueba perteneciente al módulo Generador de datos y *Playback* perteneciente al módulo Motor de ejecución. Estos subcomponentes proveen las clases que permiten la ejecución de la automatización.
1.1	La subcomponente *Recording* perteneciente al módulo Grabador de *script,* proporciona y consume las funcionalidades necesarias que permiten la captura de las actividades ejecutadas cuando se realiza la ejecución de una caso de prueba.
1.2	La subcomponente Datos de prueba perteneciente al módulo Generador de datos de prueba, permite filtrar y seleccionar los datos desde un archivo para asignar a cada uno de los elementos de la secuencia del *scripts* automatizado.
1.3	La subcomponente *Playback* perteneciente al módulo Motor de ejecución, permite reproducir los *test* de ejecución, sin intervención humana durante la ejecución
2	El paquete *Script*, representa al contendedor de todos los *scripts* automatizados y desde donde se debe extraer el archivo al que se le suministrará los datos por intermedio de la subcomponente Datos de prueba.
3	El paquete Datos, representa al contendedor de todos los archivos de datos de prueba que fueron exportados desde la BD productiva. Un ejemplo del contenido de uno de estos archivos se visualizó en la tabla 12.

Tabla 13: Descripción de componentes y paquetes.

4	El paquete *Test*, representa al contendedor de todos los *tests* de ejecución listos para ser reproducidos, todos estos archivos ya se encuentran con su información de prueba relacionada. Por intermedio de la subcomponente Playback se podrá reproducir los test de ejecución.
6	La componente Sistema de atención de clientes, se muestra como referencia al sistema que se utilizó para la ejecución de las pruebas en este segundo piloto.

Tabla 14: Descripción de componentes y paquetes.

En la tabla 14, se proporciona una descripción detallada de las clases de cada uno de los subcomponentes que se muestran en la imagen 23.

N°	Descripción
1.1	Subcomponente *Recording*
1.1.1	La clase Form contiene las definiciones de objetos, acciones y lógica del módulo Grabador de *script.* También contiene las definiciones de la GUI del sistema de atención de clientes, por ejemplo, el tamaño de los formularios que se deben abrir con la componente de Selenium WebDriver. Importante mencionar que la clase Form está configurada para interactuar con otras aplicaciones de prueba de la compañía de TV, bastaría con configurar estas nuevas aplicaciones.
1.1.2	La clase XmlDocumentTools contiene la librería de métodos para capturar las secuencias que se realizan al grabar los CP, las que se transforman en estructuras con formato XML.
1.2	Subcomponente Datos de prueba
1.2.1	La clase Main contiene las definiciones de la GUI del módulo generador de datos de prueba*.* También captura y proporciona los nombres de los objetos contenidos en los *scripts* automatizados y los nombres de los campos contenido en el archivo de datos de prueba.
1.2.2	Tools que contiene todas las funciones y métodos para ser utilizadas por los otros elementos del módulo generador de datos de prueba y permitir la asignación de los datos de prueba a los elementos del *script* automatizado
1.2.3	La clase WebCommand permite obtener los atributos de los objetos del *script* automatizado, por ejemplo: label, checkbox, radiogroup, input, etc. A los cuales se les asignará un valor.
1.2.4	La clase XmlSecuencia permite asignar los datos a cada uno de los nodos XML de la estructura de los objetos del *script* automatizado.

Tabla 15: Descripción de clases de los subcomponentes.

1.3	Subcomponente *Playback*
1.3.1	La clase Program dirige la ejecución de las pruebas automatizadas e inicializa a las otras clases para la ejecución de los *scripts*. Es decir, permite dirigir, invocando los métodos y funciones de las clases para la reproducción de un *test* de ejecución.
1.3.2	La clase Tools contiene todas las funciones y métodos utilizadas para la interactividad entre la aplicación a probar y el navegador IE, en otras palabras, es la encargada de buscar, validar, asignar valores o cualquier tipo de interacción sobre los objetos o elementos de los formularios contenidos en la aplicación web. Además, esta clase realiza la escritura de la generación del reporte de ejecución.
1.3.3	La clase Log permite escribir el paso a paso de la reproducción de un *script* automatizado. Esta clase contiene todas las funciones y métodos que utiliza la clase Tools para registrar el resultado al finalizar una tarea.
1.3.4	La clase *Scripts* contiene la lógica de ejecución de cada caso de prueba creado en la automatización de pruebas del sistema de atención de clientes.

Tabla 14: Descripción de clases de los subcomponentes.

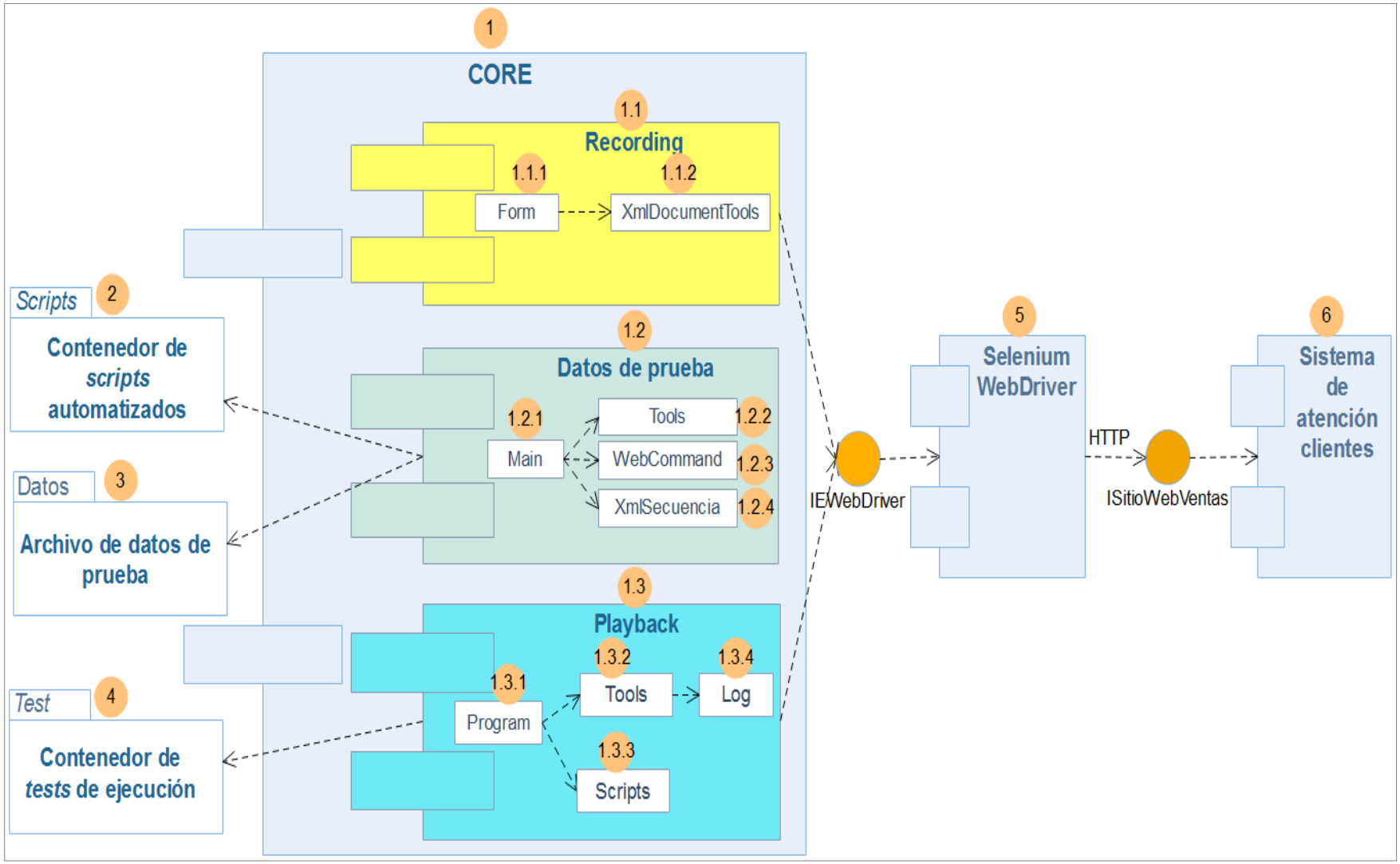

Imagen 23: Diagrama de los componentes para el prototipo.

5.5. Descripción de colaboración entre los objetos

En las siguientes secciones se muestran los diagramas de secuencias de los objetos más relevantes de cada uno de los módulos que conforman la solución, para proporcionar una mayor comprensión de la solución de automatización en su globalidad.

5.5.1. Descripción de colaboración entre los objetos al grabar los *scripts*

En la imagen 24, se presenta la secuencia de comunicación entre los principales objetos y componentes al momento de realizar la ejecución de un caso de prueba para transformar en un *scripts* automatizado de prueba.

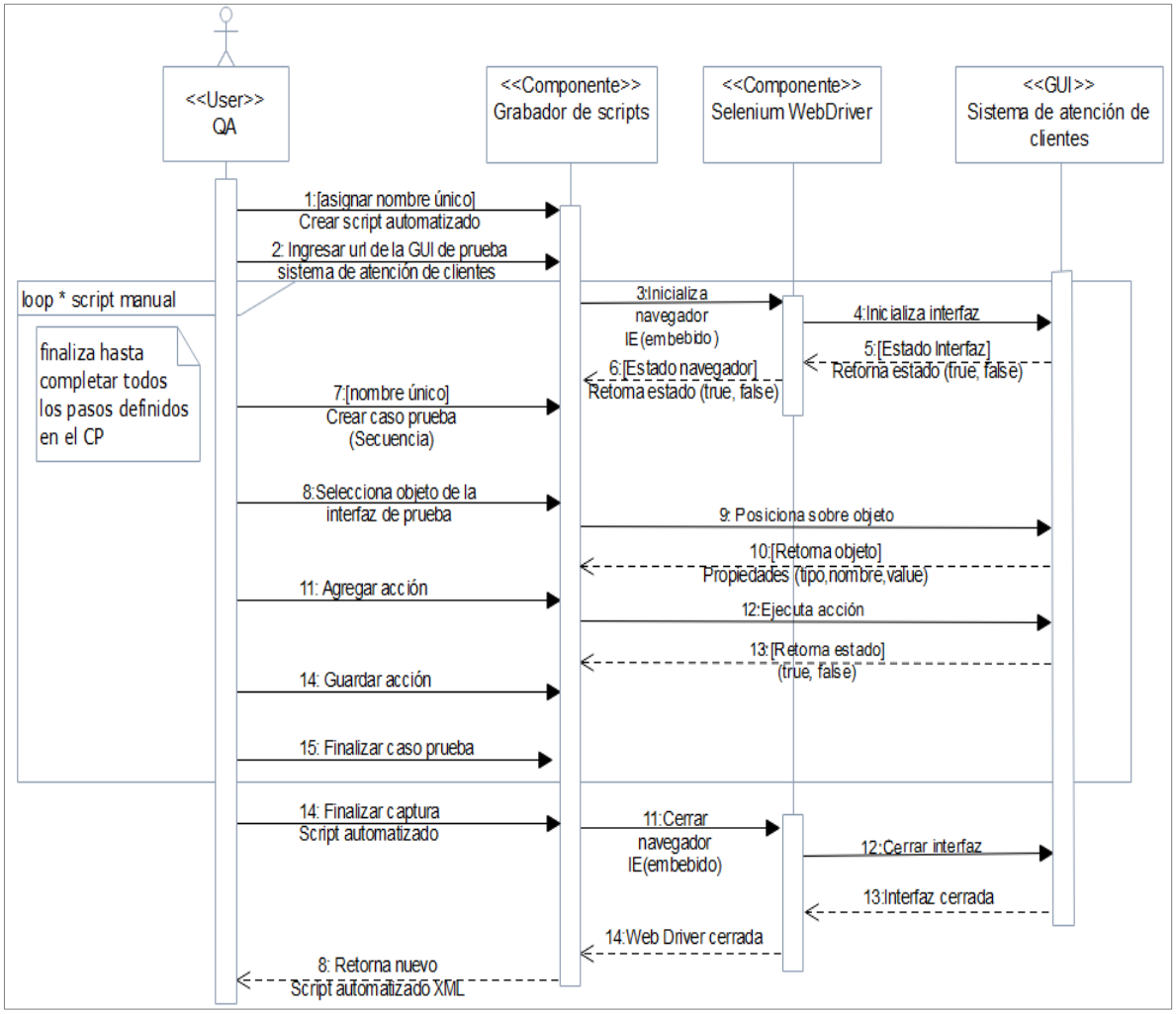

Imagen 24: Diagrama de secuencias para grabar un *script* de prueba.

El usuario QA, a través del módulo grabador de *scripts*, captura las secuencias ejecutando el procedimiento descrito en el *script* que fue confeccionado de forma manual. Cuando se inicia la captura, se debe asignar un nombre único al *script* automatizado, este se obtiene desde la matriz de trazabilidad funcional para identificar posteriormente al CP. La captura termina hasta ejecutar todos los pasos descritos, transformando el *script* manual en un *script* automatizado

Se utilizó el objeto Selenium WebDriver para embeber el navegador web IE que permitirá la interacción con la interfaz de la aplicación.

5.5.2. Descripción de colaboración entre los objetos al asignar los datos de prueba

En la imagen 25, se presenta la secuencia de comunicación entre los principales objetos y componentes al momento de generar los datos de prueba que se asignarán a los *scripts* automatizados.

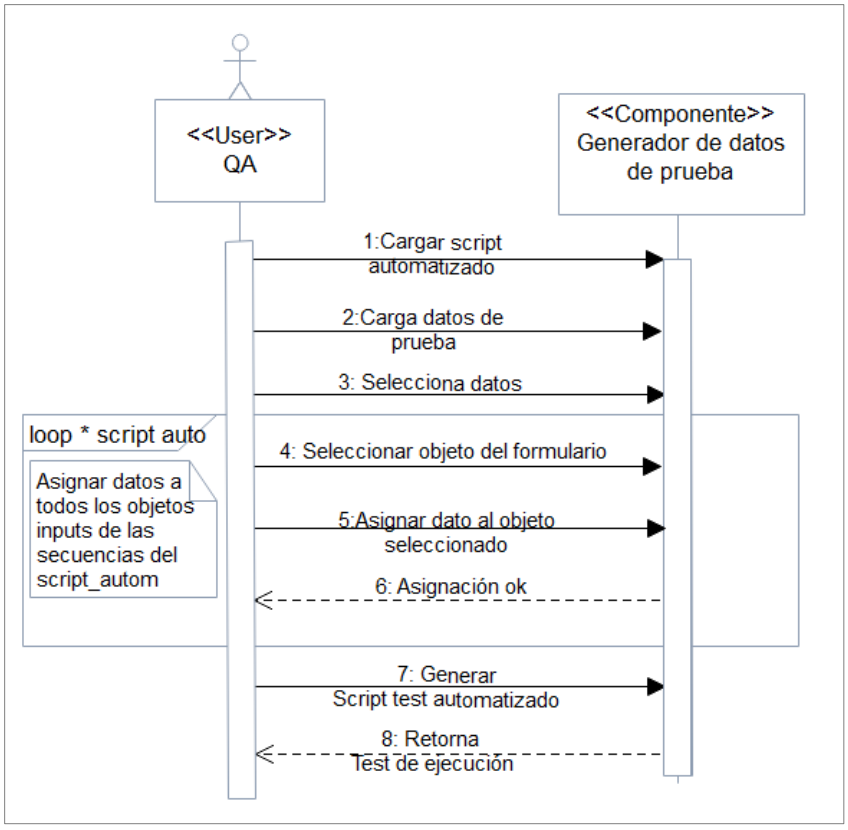

Imagen 25: Diagrama de secuencias para asignar datos de prueba.

El usuario QA, a través del módulo generador de datos de prueba, deberá cargar el *script* automatizado y el archivo que contiene los datos de prueba, pudiendo seleccionar un subconjunto de datos para asignar al *script* automatizado. Es decir,

se pude asignar varios registros a una misma prueba la que permitirá realizar, por ejemplo, pruebas con valores límites.

El proceso de asignación termina hasta que sean cubiertos todos los input del *script* automatizado, lo cual, convierte este *script* automatizado en un *script* de test ejecutable, que se podrá reproducir posteriormente a través del módulo motor de ejecución.

5.5.3. Descripción de colaboración entre los objetos para ejecutar los *scripts*

En la imagen 26, se presenta la secuencia de comunicación entre los distintos objetos al momento de la reproducción de las pruebas de regresión.

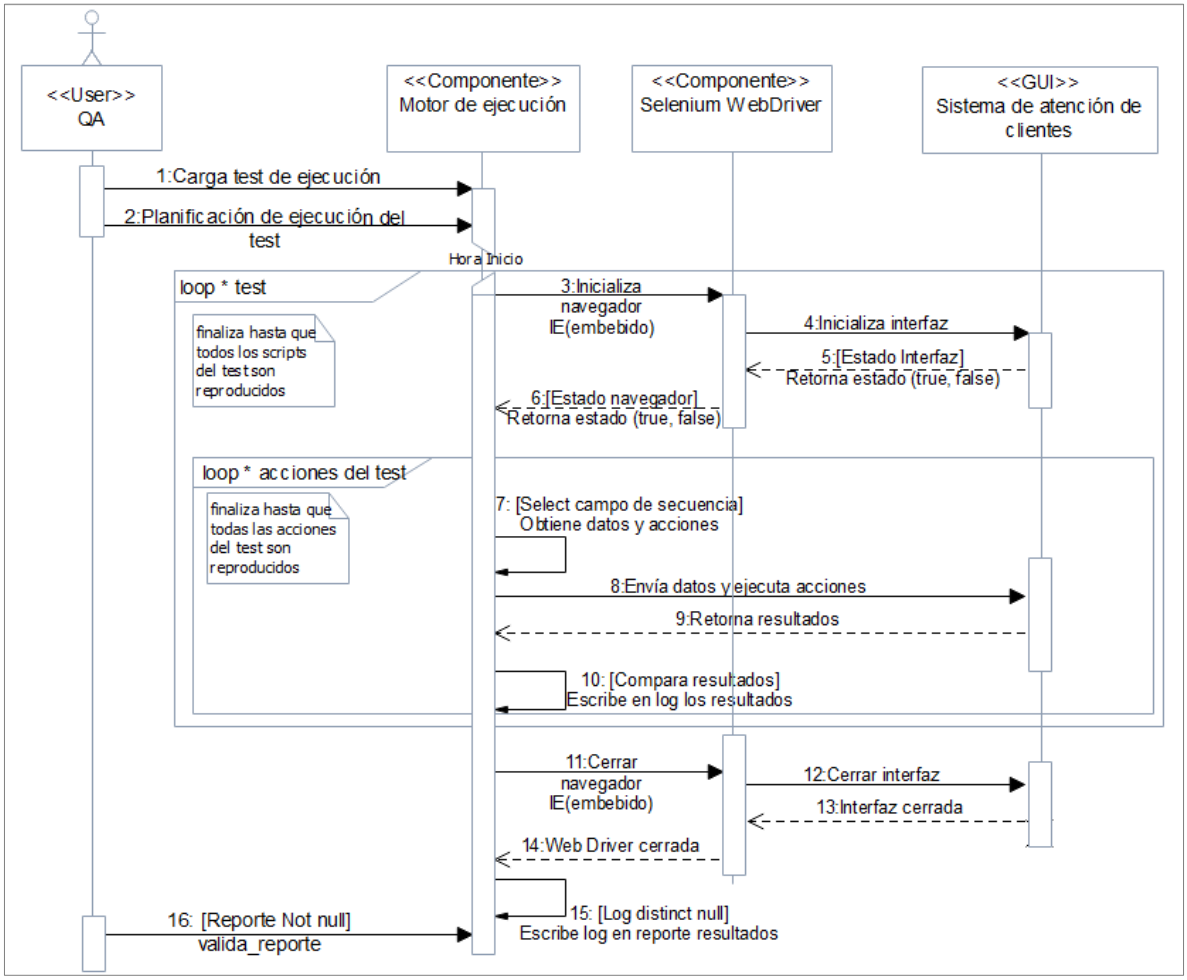

Imagen 26: Diagrama de secuencias para reproducir los *scripts*.

El usuario QA, a través del módulo motor de ejecución, deberá seleccionar los *scripts* de *tests* ejecutables y asignar fecha/hora en la que se ejecutarán. Este

motor reproducirá los pasos de las secuencias tantas veces como sea necesario hasta que los *scripts* de test sean ejecutados en su totalidad, dejando registro del paso a paso realizado para evaluar posteriormente la completitud de ejecución a través del reporte creado.

Se utilizó el objeto Selenium WebDriver para embeber el navegador web IE que permitirá la interacción con la interfaz de la aplicación.

5.6. Construcción

En esta sección se presenta una breve explicación de la metodología seleccionada para el desarrollo del prototipo y las herramientas utilizadas para poder realizar la automatización de los casos de prueba.

5.6.1. Metodología utilizada

Para el desarrollo de los módulos sistémicos de este segundo prototipo se siguió utilizando la metodología iterativa incremental, al igual que el prototipo anterior presentado en el capítulo 4. Cada incremento correspondía al desarrollo de un nuevo módulo establecido en el diseño. Cuando todos los módulos fueron desarrollados se comenzó a realizar la automatización de los CP. También se definieron iteraciones para la automatización de los *scripts* de test para incorporarlos en un nuevo proceso definido y ser utilizados como parte de las pruebas de regresión.

En cada una de las 3 iteraciones se ejecutaron las actividades: diseño manual de los *scripts*, ejecución manual de los *scripts*, grabado de los *scripts*, asignación de datos de pruebas y ejecución a los *scripts* automatizados. Para el diseño manual de los *scripts* se escribió el paso a paso de la secuencia de las pruebas, tal como se mostró en el ejemplo de la Sección 4.3.1. En la ejecución manual, se valida la secuencia escrita para comparar el resultado esperado y el resultado obtenido. Una vez efectuada la validación, se procede a grabar las pruebas generando como resultado los *scripts* automatizados. Cada iteración finaliza con la asignación de datos y ejecución automática de los test, validando los resultados al igual que la ejecución manual antes realizada.

5.7. Implementación

En esta sección de implementación se presenta la última modificación al proceso de desarrollo. Específicamente los cambios ocurrieron en el subproceso de soporte productivo, los que se presentan en la imagen 27 de color café claro.

La modificación de este subproceso, tiene como objetivo, asegurar la actualización o inserción de nuevos *scripts* automatizados en la medida que se van realizando

cambios en el sistema de atención de clientes, y en esta etapa del proceso, no se interfiere en los tiempos de implementación.

Las modificaciones que se incorporaron antes en el subproceso de *testing* en la Sección 3.2.4, se mantendrán igual para la realización de las pruebas para este prototipo.

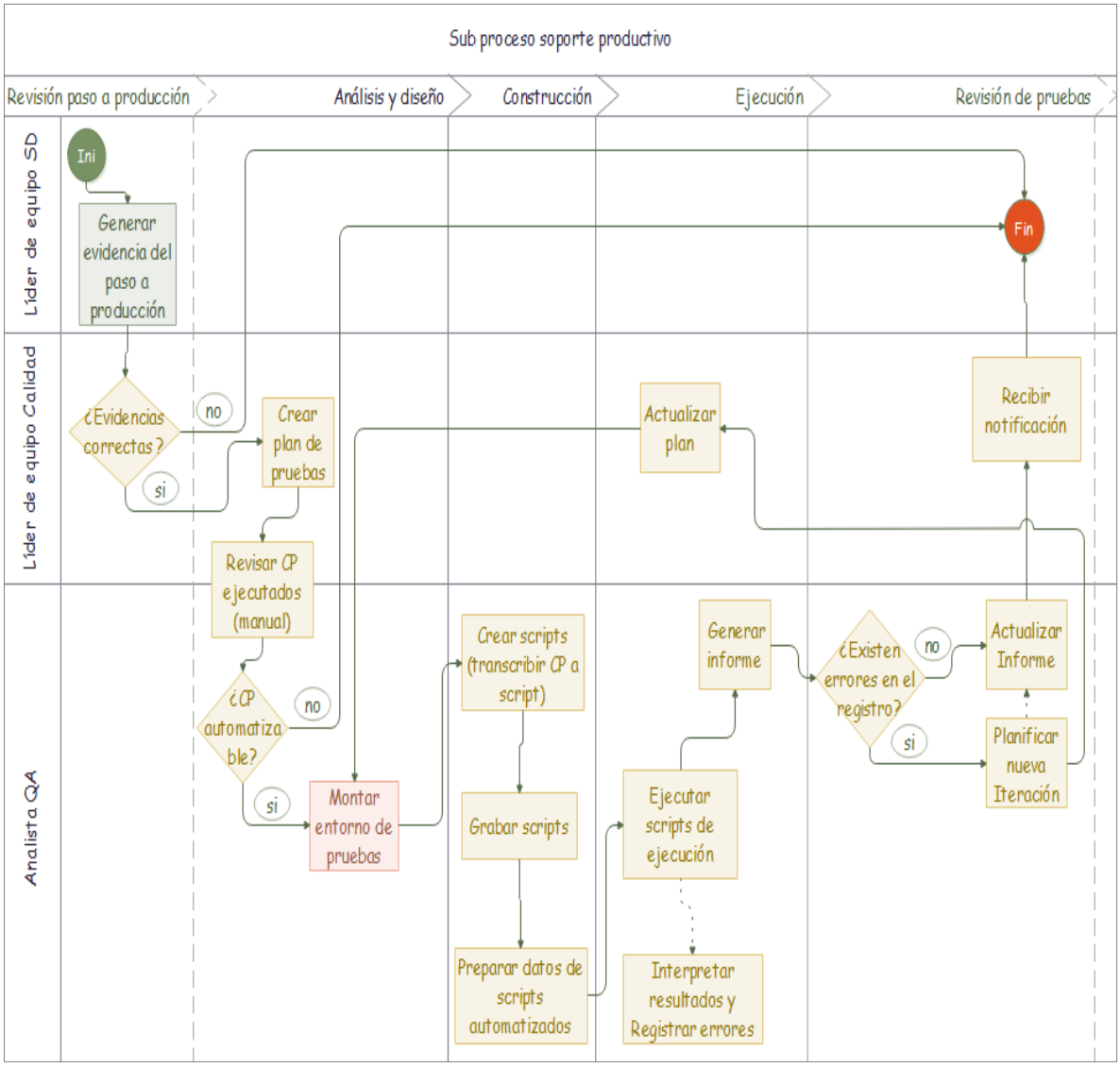

Imagen 27: Nuevo proceso de *testing* para etapa de transición.

Como se puede ver en la imagen 27, se realizaron varias alteraciones del subproceso soporte productivo, el cual contenía anteriormente solo una actividad de proceso, la cual era generar evidencia del paso a producción. A este proceso se agregaron nuevas actividades, estas son: crear un plan de prueba de

automatización, analizar los CP ejecutados de forma manual para decidir cuales se podrán automatizar bajo los criterios expuestos en las etapas de análisis, grabar, asignar datos de prueba y reproducir los nuevos *scripts*.

El plan de prueba es entregado al Analista QA, para realizar la actividad de convertir los CP a *script* automatizados, siguiendo la secuencia: transcribir los CP en *scripts* y grabar los *scripts*. Para comprobar su correcto comportamiento se sigue la secuencia: preparar datos de *script* automatizados y ejecución. Los resultados son revisados posteriormente para mantener informado al Líder de calidad TI y a su vez, mantenga actualizado el plan en caso que se tenga que volver a iterar. Las transformaciones que resultan correctas pasan a formar parte de las pruebas de regresión del sistema.

Como se indicó, este nuevo procedimiento permitirá mantener actualizados los *scripts* automatizados de funcionalidades productivas del sistema de atención de clientes producto de cambios o nuevas funcionalidades de este sistema.

5.8. Propuesta de mejora

Este segundo prototipo fue construido para proporcionar agilidad al proceso de *testing* del sistema de atención de clientes y para detectar posibles errores a pruebas de regresión que no se venían haciendo por falta de recursos (tiempo y personas) cada vez que se realizaba un nuevo release de este sistema.

Esta solución permitió entregar una herramienta robusta entregando mensajes claros frente a situaciones de anormalidad. También se abordó en la solución las oportunidades de mejora detectadas en el primer prototipo. Esta herramienta permite automatizar casos de prueba sin tener que codificarlos por un programador, esto reduce el tiempo en el proceso de automatización de los *scripts* y también permite modificarlos frente a un cambio producto de la alteración de la lógica o GUI del sistema testeado[3] que impide la ejecución correcta del *script* y en consecuencia de la acción de prueba.

Al analizar el ámbito y las características de la herramienta construida, se considera que podría a futuro incorporar en los distintos módulos construidos la conexión directa a una base de datos para realizar transacciones de alta, bajas y modificación de los *scripts*, lo cual permitiría una solución más escalable para incorporar pruebas automatizadas a otros sistemas de la compañía. Estos cambios ayudarían en la administración más segura de los *scripts* y en la eliminación de los repositorios actuales que se administran manualmente (en carpetas compartidas). Otra consideración a tener presente seria la realización de

[3]Esta palabra representa la acción de realización de pruebas sobre un sistema referido.

servicios Web (WS) per permitan la extracción directa de los datos de prueba desde ambientes homologados para estos efectos.

6. RESULTADOS DE LA IMPLEMENTACIÓN

En este capítulo se visualizan los resultados de las métricas (KPI) de los distintos indicadores que se miden dentro de TI y que se plantearon mejorar a partir del esfuerzo realizado con la implementación de los prototipos de automatización en el proceso de *testing*. Se analizan los resultados del objetivo general que se propuso y de los objetivos específicos.

6.1. Método de validación

Para la validación del objetivo principal y los objetivos específicos que se propusieron mejorar se utilizó las métricas (KPI) que fueron definidas por el gerente senior de TI a principio de año, las cuales, son revisadas en las reuniones mensuales de la gerencia. Estos KPI miden la productividad de cada una de las áreas TI, estos son:

- Los KPI de service delivery manager (SDM) son: disponibilidad de la plataforma mayor al 99%, resolución de incidentes (IN) nivel 1 mayor al 85% en menos de 1 día y llamadas atendidas por mesa de ayuda mayor al 95%.
- Los KPI de solution delivery (SD) son: time to market de desarrollo menor a 65 días, HH de desarrollo evolutivo mayor al 70% del tiempo total y de desarrollo correctivo menor al 30%, resolución de incidentes (IN) nivel 2 mayor al 80% en menos de 5 días y requerimientos entregados a tiempo (en plazo) mayor al 87%.
- Los KPI de gestión de la demanda (GD) son: time to market de evaluación de requerimientos menor a 15 días, backlog menor a 40 iniciativas y antigüedad del backlog menor a 80 días.

Todas estos KPI son obtenidas desde las herramientas IBM Control Desk para la administración de ticket y Microsoft Project & Portfolio Management (PPM) para la administración de proyectos y requerimientos.

En la validación de ambos prototipos participaron:

- Gerente de SD, en la toma de decisión respecto a la implementación en el proceso.
- Líder de calidad productiva, involucrado en el análisis, diseño y desarrollo del proyecto. Además de automatización de *scripts*.
- Un programador experto que ayudo en la construcción de la herramienta de automatización y en la automatización de *scripts*.
- Un analista QA, en la generación de *scripts* manual y automatización de los mismos en el segundo prototipo.

Estos 3 últimos usaron directamente las herramientas construidas de ambos prototipos.

El proceso que se siguió para la validación fue el siguiente:

- Se recopilo información anterior a septiembre 2015 de los paneles de control, información proporcionada por el Gerente SD, el Gerente SDM y jefe de GD.
- Se hizo seguimiento especial de los datos presentados en las reuniones de paneles de control a partir de septiembre 2015 en adelante, fecha que se lanzó el primer prototipo.
- Se realizaron reuniones periódicas, posterior al panel de control, con el Gerente de SD, el equipo de calidad productiva (Líder, analistas QA) y programador para evaluar la implementación de los prototipos según fueron ocurriendo, donde se analizaron: tiempos del proceso de automatización, facilidad de mantenimiento, facilidad de uso y expectativas. En estas reuniones se hicieron preguntas abiertas que permitieron al Gerente de SD determinar la difusión y los cambios en el proceso de *testing*.
  Preguntas planteadas:
  1. ¿Considera usted que el prototipo de automatización de pruebas proporciona la agilidad necesaria para ser incorporado al proceso de *testing* de punta a punta?
  2. ¿Considera usted que el prototipo de automatización de pruebas proporciona la flexibilidad de mantenimiento necesaria para ser incorporado al proceso de *testing* de punta a punta?
  3. ¿Considera usted que el prototipo de automatización de pruebas, cubre su principal función, la de entregar software de mejor calidad en los sistemas piloteados?
  4. De acuerdo a su percepción, en una escala de 1 (poco usable) y 10 (muy usable), ¿Cómo calificaría usted al prototipo de automatización de pruebas?, esta pregunta se hizo cuando ya se habían realizado 3 pasos a producción, en los cuales se hizo pruebas de regresión usando la herramienta.
  5. ¿Qué funcionalidades le parecieron que debe ser mejoradas en una próxima versión?

6.2. Resultados obtenidos

Como objetivo general se propuso; definir, formalizar e instrumentar proceso de *testing* enfocado en la automatización de las pruebas de regresión para entregar software de mejor calidad en la compañía de TV Chile. Al respecto, se logró en una etapa de transición, modificar el proceso de *testing* de acuerdo a lo citado en la sección 4.6, para finalizar con la implementación del proceso presentado en la sección 5.6.

No obstante, la propuesta de entregar software de mejor calidad, fue también conseguido en los sistemas participantes de los prototipos como pilotos de prueba (ventas y atención de clientes); estos redujeron el número de incidentes de acuerdo a lo presentando en la imagen 28, donde se puede apreciar estos 2 sistemas evaluados. En el mes de septiembre del 2015 existió una importante reducción a la mitad en comparación al mes anterior. También se visualiza en abril del año 2016, una reducción de incidentes con la implementación del segundo prototipo que abordo pruebas automatizadas al sistema de atención de clientes. Desde abril de este año, se logró una estabilidad en estos sistemas y los errores detectados en producción corresponden mayoritariamente a problemas en las nuevas funcionalidades que son probadas de forma manual.

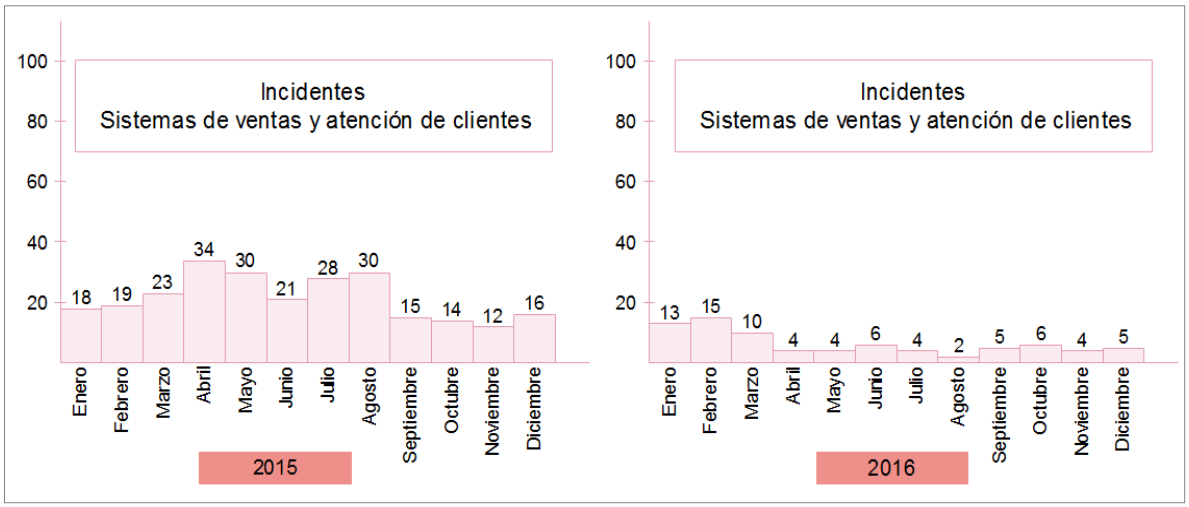

Imagen 28: Representa los incidentes de los sistemas piloteados en 2015 y 2016.

Como objetivos específicos se propuso:

- Reducir el Time to market a 65 días; al realizar pruebas automatizadas de regresión en los sistemas de ventas y el sistema de atención de clientes, permitió realizar validaciones en menor tiempo. Como se visualiza en la tabla 15, la métrica Time to market se redujo y permitió finalizar el año 2015 con 61 días y en 55 días el año 2016.

			2015													
KPI	Medida	SLA	Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sep	Oct	Nov	Dic	Promedio
Aging del backlog	Días	<=80			210	135	116	118	111	133	68	78	42	70	**108**
Time to market	Días	<=65			43	80	49	49	96	36	68	90	63	32	**61**
# Backlog	Nro	<=40	39	43	45	56	51	53	37	34	45	32	34	30	**42**

			2016												
KPI	Medida	SLA	Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sep	Oct	Nov	Dic	Promedio
Aging del backlog	Días	<=80	65	61	51	60	65	58	59	70	65	62	60	68	**62**
Time to market	Días	<=65	43	69	52	66	60	56	62	54	45	47	51	60	**55**
# Backlog	Nro	<=40	28	31	45	40	38	37	28	29	29	35	38	29	**34**

Tabla 16: Los 3 indicadores más importantes de TI.

- Reducir al 10% las HH de pruebas manuales de los equipos de SD en comparación del esfuerzo total del requerimiento. La realización de pruebas automatizadas de regresión en los sistemas utilizados en los prototipos, permitió realizar una mayor cobertura de pruebas utilizando menos tiempo que un proceso manual. Sin embargo, no se logró la reducción de tiempo esperado, solo se alcanzó reducir al 17%, de acuerdo al siguiente análisis que se presenta en los siguientes párrafos.

  Anteriormente, con el proceso manual, se destinaba el 30% del tiempo total por requerimiento, esto implicaba mensualmente un costo de 60 HH de las 200 HH destinadas al desarrollo de los requerimientos evolutivos en el sistema de ventas y de 108 HH del total de 360 HH en sistema de atención de clientes.

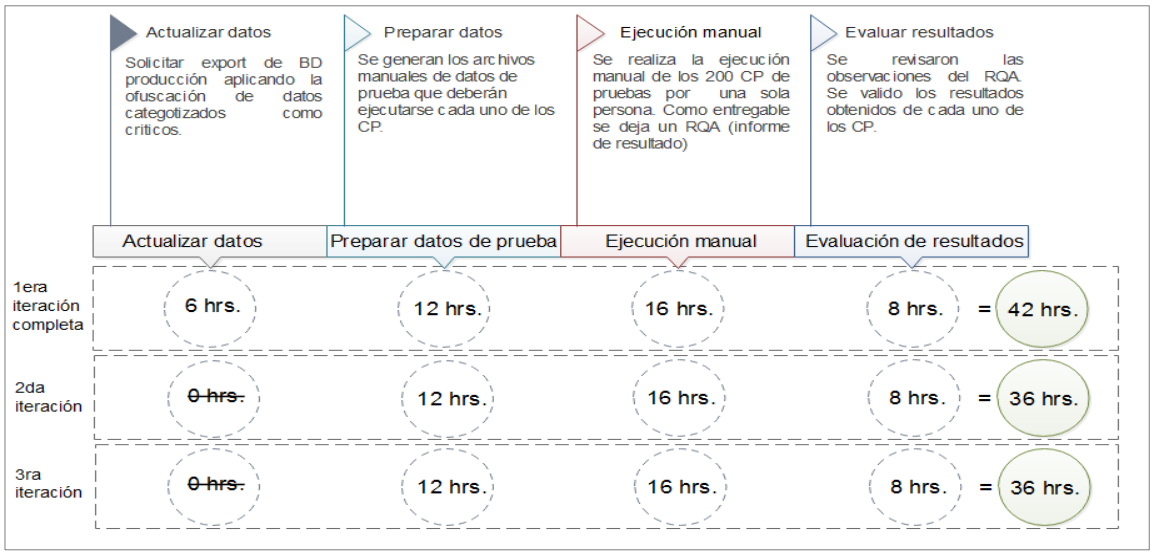

Imagen 29: Tiempos de evaluación ejecución manual.

En la imagen 29 y 30 se realiza la comparación del tiempo manual de ejecución versus tiempo de ejecución automatizado para el sistema de atención de clientes. Se puede apreciar, el tiempo total para la ejecución manual de los casos de prueba tiene un costo de 42 horas, de encontrar fallos y se quisiera volver a chequear en una segunda iteración, se debe invertir otras 36 horas más en cada nueva iteración hasta que se logre la validación sin errores; sin embargo, esta misma validación de forma automatizada de acuerdo a la imagen 30 requiere de:

- 38 horas total, proceso completo, incluido actualizar y volver a grabar los casos de prueba que fueron modificados por actualizaciones en el sistema de atención de clientes. Los pasos grabar *scripts* y preparar datos (8 horas y 4 horas) pueden variar en tiempo, dependiendo de los cambios que se deban ejecutar en los *scripts*, generalmente son mucho menos.
- 14 horas total, desde la segunda iteración para ejecutar el 100% de las pruebas; esta reducción del 63% del tiempo es muy significativo cuando se requiere volver a probar todo nuevamente, lo que permite dentro de un solo día hacer el proceso completo de pruebas. El proceso de ejecución puede hacerse durante la noche (sin intervención humana), el cual tarda 12 horas y 2 horas más para revisar los resultados al siguiente día.

Generalmente, en las pruebas se alcanzan a desarrollar hasta 3 iteraciones, lo que equivale a 114 horas en pruebas manuales (42 horas en iteración 1 + 36 horas en iteración 2 + 36 horas en iteración 3). Con la automatización de las pruebas, se logra realizar en 64 horas (38 horas en iteración 1 + 14 horas en iteración 2 + 12 horas en iteración 3), reduciendo, el proceso de prueba al 17% del total de horas disponibles.

- Reducir a la mitad las HH de UAT utilizando a los usuarios de negocio; antes de la automatización se solicitaban 4 usuarios para la certificación de nuevas funcionalidades y para la validación de las pruebas de regresión; hoy solo se utiliza 1 usuario para la certificación de las nuevas funcionalidades, ya que las pruebas de regresión se hace con la herramienta de automatización invirtiendo tiempo solo en la etapa final de evaluación de resultados entre 2 a 8 horas como se visualiza en la imagen 30.

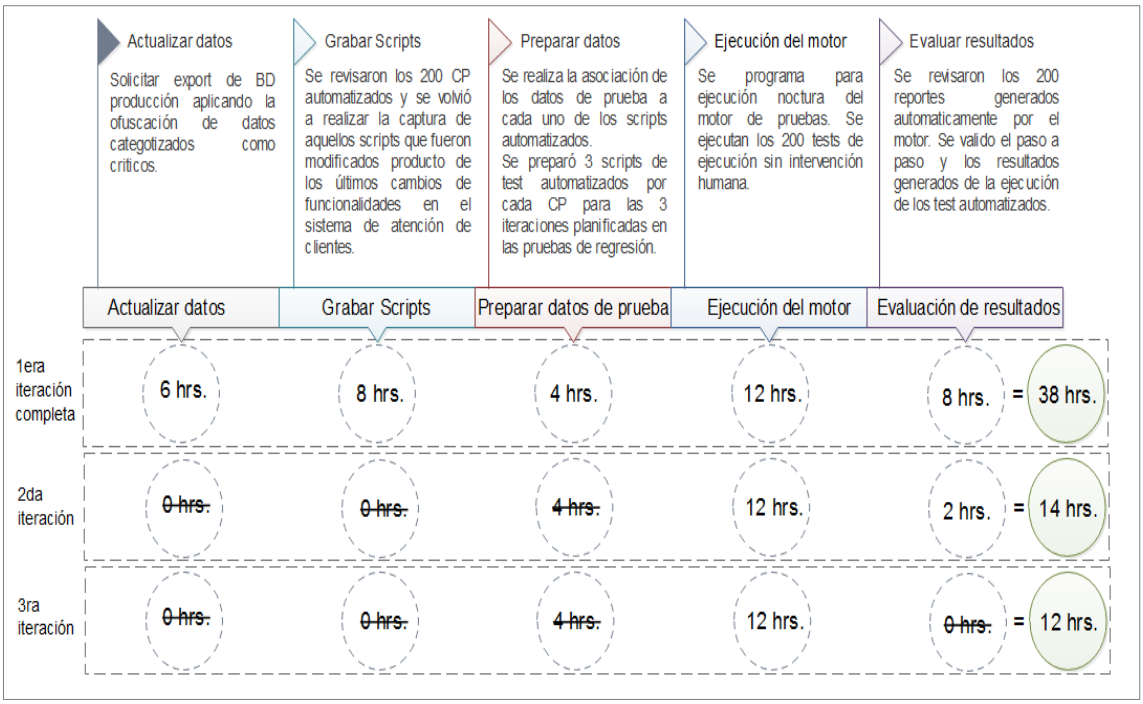

Imagen 30: Tiempos de evaluación ejecución automatizada.

- Automatizar más del 65% de los casos de prueba de cada sistema.

    - En el caso del sistema de ventas, la cantidad de casos de pruebas que se identificaron producto de este trabajo, fue de 169 casos; 99 casos de prueba más de los inicialmente considerados cuando se ejecutaban en forma manual (solo se ejecutaban cerca de 70 casos de prueba), y el objetivo propuesto fue automatizar 50 casos de prueba. Se automatizó en el primer prototipo los 50 casos y con la versión nueva del segundo prototipo se alcanzaron a 115 casos de prueba en total.
    - En el caso del sistema de atención de clientes, la cantidad de casos de prueba que se identificaron fue de 296, y el objetivo propuesto fue automatizar 200 casos de prueba, logrando automatizar estos 200 casos de pruebas.
- Reducir las HH invertidas para mantenciones correctivas. En la tabla 16 obtenida del panel de control de cierre mensual de marzo 2016, se puede apreciar que a esta fecha, solo en el sistema de atención de clientes se logró el objetivo de reducir a menos del 30% de utilización del volumen de horas disponible mensualmente para mantenciones correctivas.

Aplicación	Medida	Capacidad mensual del equipo	Incidentes	HH Corrección	% Tiempo mantención correctiva
Sistema de ventas	HH	200	4	80	40%
Sistema de atención de clientes	HH	360	2	40	11%

Tabla 17: Porcentaje de tiempo ocupado en mantenciones correctivas.

7. CONCLUSIONES

En este capítulo final se plasman las conclusiones de la realización del proyecto ejecutado a través de los dos prototipos y también la conclusión personal respecto de la valorización de este trabajo. Se finaliza con algunas recomendaciones futuras.

7.1. Conclusiones del proyecto

La realización de ambos prototipos permitió la depuración de los CP de cada uno de los sistemas que se pilotearon, eliminando aquellos que resultaban redundantes y ampliar con nuevos CP aquellas funcionalidades no cubiertas. Lo anterior permitió al equipo de Calidad productiva, específicamente a las personas que realizan el rol de QA, adoptar prácticas enriquecedoras para la formulación de CP y ampliar a otros sistemas la generación de los mismos.

Respecto a la metodología utilizada, en el primer prototipo, esta permitió mejorar el tiempo del proceso de prueba, al entregar los *script* automatizados en la medida que se iban terminando las iteraciones, lo cual permitió ejecutar un proceso mixto de prueba en la etapa de transición, por una parte se utilizaron estos *script* automatizados combinado con las pruebas manuales de los otros CP requeridos en la pruebas de regresión. Sin embargo, esta mejora del tiempo del proceso de prueba se vio afectada por la arquitectura definida, al requerir varias horas de programación para transformar un *script* de prueba en un *script* automatizado y cualquier cambio en el sistema utilizado en el prototipo como piloto requería también varias horas para realizar las modificaciones en los *scripts* ya automatizados.

Por el contrario, la metodología del segundo prototipo no permitió utilizar CP automatizados para ocupar prontamente en algún proceso de prueba, ya que se requería terminar los 3 módulos para recién poder comenzar con la automatización de los *scripts*. Mientras que la arquitectura definida permitió en pocos minutos transformar un *script* de prueba en un *script* automatizado, permitiendo además mantener actualizados los *scripts* automatizados en paralelo a cómo van ocurriendo los cambios en el sistema utilizado como piloto.

Como se visualizó en la imagen 30, el tiempo completo para actualizar y preparar un proceso de prueba de 200 CP requiere de 12 HH (8 horas para actualizar los *scripts* + 4 horas de preparación de datos de prueba). Y otras 12 horas de máquina para la reproducción de los 200 *test*, cuya ejecución es desatendida (sin intervención humana), comparada con la ejecución manual presentada en la imagen 29, de 16 horas de una persona dedicada.

Los equipos de desarrollo evidenciaron también mejoras en sus entregables, integrándose en etapas previas con el equipo de calidad para ejecutar pruebas (no formales) y que les permita evidenciar posibles errores.

Respecto al proceso final de desarrollo, donde se intervino los subprocesos de *testing* y soporte productivo, donde la combinación de buenos probadores en las pruebas manuales para la certificación de nuevas funcionalidades junto con el soporte de una herramienta de ejecución automatizada para las pruebas de regresión permitió aumentar la eficacia y velocidad de forma sorprendente. Esta combinación de pruebas exploratorias manuales y navegación automatizada resultan muy eficaz para el proceso en general.

Respecto a los plazos, este proyecto en su globalidad superó en más de la mitad al establecido al iniciar el proyecto (planificado = 6 meses, real 12 meses). El esfuerzo de llevar a cabo, capturar las interacciones realizadas en la aplicación satélite y su posterior reproducción automatizada sin que exista codificación de un programador especialista, requirió de mucho más tiempo respecto a la planificación inicial considerada para el proyecto de tesis.

Este proyecto permitió reducir los errores de dos sistemas satélites importantes de la compañía de TV, los cuales se utilizan para atender a clientes de la compañía y cualquier error que se produzca impide en una buena atención de los clientes, al no poder gestionar correctamente sus solicitudes y en consecuencia una mala evaluación del servicio que se brinda. Por lo cual, se podría concluir que la realización del proyecto ayuda a la compañía de TV en mantener buenos niveles de satisfacción de los clientes.

7.2. Conclusiones personales

Años atrás lideré un proyecto de mejora CMMI nivel 2 en una empresa de desarrollo de software; proyecto que no se logró implementar en la organización. De este proyecto se rescató una lección muy importante: Para realizar iniciativas de mejora es necesario contar con el apoyo incondicional de los niveles superiores de la organización.

Antes de iniciar este proyecto de tesis, se realizó un diagnostico que permitió evidenciar los problemas y obtener el patrocinio de la gerencia TI, con el que se definieron los objetivos alineados a los beneficios que se pueden alcanzar con la automatización en las pruebas. Lo anterior permitió trabajar en este proyecto como una iniciativa de mejora impulsada por el área de calidad, en la cual particularmente me encontraba al iniciar este proyecto.

La automatización de casos de prueba de regresión, me permitió abordar en detalle temáticas en las que no se tenía experiencia en un ámbito laboral. El conocimiento adquirido durante los cursos: Introducción a la gestión de calidad de

software, Técnicas de prueba de software, y Métricas y Calidad del software, fueron fundamental para tener una mirada distinta respecto a cómo veníamos realizando las pruebas de software.

Todos los cambios al interior de las organizaciones producen conflictos que deben ser gestionados como parte del proyecto, fue fundamental para lograr concluir el desarrollo e implementación de los prototipos los cursos: Negociación de proyectos TI, y Liderazgo y trabajo en equipo, los que me permitieron en momentos críticos lograr acuerdos y hacer que el proyecto sea un trabajo de equipo sumando a otros miembros de TI para apoyar la iniciativa.

En resumen, la realización del proyecto de automatización de pruebas de regresión permitió volcar el conocimiento adquirido en el plan de estudio de los distintos ramos cursados en el Magíster en Tecnologías de Información.

Como lecciones aprendidas para un próximo proyecto de estas características, rescato 3 puntos importantes que se deben considerar:

- Establecer objetivos realistas y proporcionar recursos suficientes para lograr el retorno previsto sobre la inversión.
- Ser demasiado ambicioso puede dañar los esfuerzos de automatización e impedir beneficios reales y de menor escala; en el caso del último prototipo, existió instancias de cancelar el proyecto. El esfuerzo realizado fue superior al planificado inicialmente y siendo una iniciada de proyecto de tesis se asignó mayor tiempo y que finalmente permitió proseguir para conseguir la herramienta válida que nos permitiría abordar las pruebas automatizadas.
- Una buena arquitectura debe tener los niveles adecuados de abstracción para dar flexibilidad y capacidad de adaptación reduciendo los costos de mantenimiento de todos los aspectos de la automatización.

7.3. Recomendaciones a futuro

Dado el amplio espectro y alcance que puede tener este tipo de herramienta en la automatización de pruebas y en la automatización de procesos asistidos por personas, se consideran entre las más importantes, las siguientes propuestas de extensión a la herramienta y a otros escenarios de la compañía de TV.

- De acuerdo a los resultados presentados en el capítulo 6, se recomienda seguir masificando la automatización de las pruebas de regresión para el resto de los sistemas satélites, con el fin de entregar software de mejor calidad.
- Se recomienda extender las prueba de automatización no solo para grabar las acciones del usuario en la interfaz gráfica, también a establecer formalmente pruebas con las entidades que se integran a los sistemas como son: bancos y comercios.

- Se recomienda evaluar adquirir o desarrollar una herramienta que permita gestionar la administración de las pruebas, sobre todo si se atiende la recomendación de extender a otros sistemas de la compañía. Esto facilitará la gestión de los casos de prueba y su trazabilidad funcional.
- Se recomienda extender la automatización a otros escenarios propios del negocio de la compañía de TV, por ejemplo: auto atención del cliente por intermedio de SMS, IVR telefónico en funcionalidades como la contratación de productos adicionales, compra de PPV, activación de clientes, entre otras. Estas funciones vienes operadas o asistidas por ejecutivos de atención, con un alto costo de infraestructura, la automatización de estos servicios podría reducir el volumen de llamados y aumentar la satisfacción de los clientes al no tener que esperar ser atendidos por una persona para este tipo de solicitudes que perfectamente pueden ser automatizadas.

BIBLIOGRAFÍA

[Buwalda, 2011] Hans Buwalda, “Action Based Testing” magazine, A Modularized Keyword Approach, february 2011.

[Copeland, 2015] Copeland, Lee, *What Are the Key Components of an Effective Test Strategy?*, [en línea], [Consulta: 08.marzo 2016] <https://www.stickyminds.com/>.

[Gheorghiu, 2005] Gheorghiu, Grig., “A Look at Selenium”, Publisher https://www.stickyminds.com/better-software-magazine/look-selenium.

[Graham, 2012] D. Graham and M. Fewster, “Reflections on the Case Studies,” in Experiences of Test Automation: Case Studies of Software Test Automation, Addison-Wesley Professional of Publisher, 2012.

[Hagar, 2014] Hagar, Jon, *How to Design a Test Strategy* [en línea], [Consulta: 08.marzo 2016] <https://www.stickyminds.com/>.

[Hayes, 1996] Hayes, Linda G., “Automated testing handbook”, first edition, software testing institute of Publisher 1996.

[Jorgensen, 2013] Jorgensen, Paul C., “Software Testing”, 4th edition, Auerbach Publications of Publisher 2013.

[Madi, 2013] R. Madi, “Automating Functional Tests” in Learning Software Testing with Test Studio, Packt Publishing, 2013,ch 2.

[McKay, 2014] Judy McKay, Graham Bath, “The Software Test Engineer's Handbook”, 2nd edition, Publications of Rocky Nook , 2014, ch 13.3.

[Mendoza, 2011] Sergio Mendoza Fariña, 2011. Estudio de la metodología de automatización del testeo en aplicaciones web para e-Catalunya. Memoria de Ingeniería informática. Barcelona, Universidad Politécnica de Catalunya. Facultad de Informática de Barcelona. 165p.

[Sahi, 2016] Sahi Pro, [en línea], [Consulta: 06 enero 2016] <http://sahipro.com/docs/introduction/index.html>.

[Selenium, 2016] Proyecto Selenium, [en línea], [Consulta: 06 enero 2016] <http://www.seleniumhq.org >.

[Soria, 2016] Daniel Soria Murillo, *ingeniería de software*, [en línea], [Consulta: 08 marzo 2016] <http://ingenieriadesoftware.mex.tl/61885_Modelo-V.html>.

[Spillner, 2014] A. Spillner, T. Linz and H. Schaefer, “Testing Related to Changes and Regression Testing,” in Software Testing Foundations, 4th ed., Rocky Nook of Publisher, 2014, ch 3, sec 7.4.

[StickyMinds, 2016] Comunidad StickyMinds, 2011-2017 TechWell Corp [en línea], [Consulta: 08 marzo 2016] <https://www.stickyminds.com/>.

[Watin, 2013] MinniePandey, [en línea], [Consulta: 06 enero 2016] <https://www.codeproject.com/Tips/658947/Watin-An-Automation-Testing-in-NET>

[Yépez, 2004] Maria Paulina Yépez La Rosa y Ana Gabriela Márquez, “Mejoras en los procesos de aseguramiento de la calidad en el desarrollo de aplicaciones”, trabajo de tesis, publicado 2004.

[Zambelich, 1998] Zambelich, Keith, “Totally data-driven automated testing”, http://www.sqa-test.com/articles.html.

ANEXOS

A continuación se presentan los anexos A y B. El anexo A proporciona información que permite entender el criterio de selección de los sistemas utilizados como pilotos en los prototipos de automatización y el anexo B muestra parcialmente la herramienta de automatización construido en el segundo prototipo, con el objetivo de mostrar al lector lo simple que resultaría una modificación o agregar nuevas funcionalidades del sistema que se está automatizando. Se muestran algunas imágenes de la herramienta de los distintos módulos funcionales.

A. Método utilizado para seleccionar los sistemas pilotos de los prototipos

En este anexo se indica el criterio de selección utilizado para definir los 2 sistemas satélites cuyas pruebas fueron automatizadas en cada uno de los prototipos y que permitieron pilotear las soluciones para alcanzar los objetivos planteados.

En la filial de TV Chile existen varios sistemas satélites que pudieron ser candidatos y que dan soporte a las distintas áreas de negocio, sin embargo, para lograr alcanzar los objetivos planteados era necesario seleccionar sistemas que superen algunas umbrales de indicadores importantes de TI. Se utilizaron en una primera instancia 2 indicadores relevantes, estos fueron: backlog de requerimientos pendientes e impacto en el negocio, como resultado de los más de 30 sistemas satélites analizados, se obtuvo 4 sistemas a considerar (sistema de ventas, sistema de atención de técnicos, sistema de atención de clientes y sistema call center).

Cualquiera de estos 4 sistemas eran válidos para ser parte de los prototipos, sin embargo, para la selección final se incluyeron otras variables al criterio de selección (ver tabla 17) permitiendo determinar: el primer prototipo se realizará utilizando como piloto el sistema de ventas, principalmente por la existencia de backlog pendiente y los pocos cambios que impactan en la capa de presentación (candidato ideal para procesos de automatización de pruebas), además de presentar errores en producción donde la automatización podría ayudar a cubrir en mayor porcentaje pruebas que permita detectar estos a tiempo. También se definió automatizar las pruebas en un segundo prototipo utilizando como piloto al sistema de atención de clientes, para lograr realizar una mayor cobertura de pruebas que por tiempo y capacidad no se alcanzan a realizar en forma manual.

La estrategia de realizar la automatización de pruebas de regresión mediante prototipos piloteando sistemas diferentes, tenía como objetivo abordar la solución con distintas problemáticas de negocio proporcionando al término de los prototipos una solución más robusta desde el punto de vista automatización.

Sistemas satélites	Backlog	Impacto	Horas hombre	Cambios capa presentación	Incidentes	Usuarios para test	Casos de prueba	% Ejecución	Piloto
Sistema de ventas	SI	Alto	200	Bajo	SI	2	70	<65%	✓
Sistema de atención de técnicos	SI	Alto	70	Bajo	NO	1	50	>90%	✗
Sistema de atención de clientes	SI	Alto	360	Medio	SI	4	296	<50%	✓
Sistema call center	SI	Alto	150	Alto	SI	2	95	<80%	✗

Tabla 18: Sistemas de mayor impacto con backlog de requerimientos pendientes.

Alcance de las variables para los criterios de selección:

- Backlog, representa al momento del análisis la existencia en los últimos 6 meses de requerimientos pendientes y que no pudieron ser ejecutados por falta de capacidad de horas hombre.
- Impacto, representa lo criticidad del sistema de acuerdo al número de usuarios afectados cuando este presenta alguna falla (alto mayor a 200 usuarios).
- Horas hombre, representa la cantidad de HH disponible de los equipos de SD y los centros de competencia, según corresponda para desarrollar requerimientos donde la cantidad asignada no alcanza a cubrir la demanda existente y como consecuencia la existencia del backlog.
- Cambios capa presentación, representa los requerimientos en los últimos 6 meses en cuyas necesidades afecto la capa de presentación en estos sistemas (bajo menor al 20%, medio del 20% al 50% y alto mayor al 50%).
- Usuarios para test, corresponde al número de usuarios que se solicitan al negocio para apoyar las pruebas tanto las de las nuevas funcionalidades como las de regresión para cada nueva versión.
- Casos de prueba, número de casos de prueba formales considerados como pruebas de regresión que deberían ejecutarse.
- % ejecución, corresponde a la cantidad de pruebas que se alcanza a cubrir o ejecutar en forma manual respecto del total definido (bajo menor al 65%, medio entre el 65% y 90% y alto mayor al 90% - donde bajo es más malo).

B. Herramienta de automatización construido en el segundo prototipo

En este anexo se visualiza el sistema de automatización de pruebas de regresión construido en el prototipo 2. Consta de 3 módulos que ya fueron descritos en el capítulo 5.

B.1. Módulo grabador de scripts

El módulo grabador de *scripts* permite transformar un *script* manual en *script* automatizado. En la imagen 31, se puede visualizar la GUI de este módulo, el cual

proporciona 5 funcionalidades principales para realizar la transformación del *script,* estas son:

1. Barra de dirección (URL).
2. Frame de atributos del objeto HTML.
3. Frame de atributos del objeto XML.
4. Barra de vista de los objetos.
5. Browser de navegación.

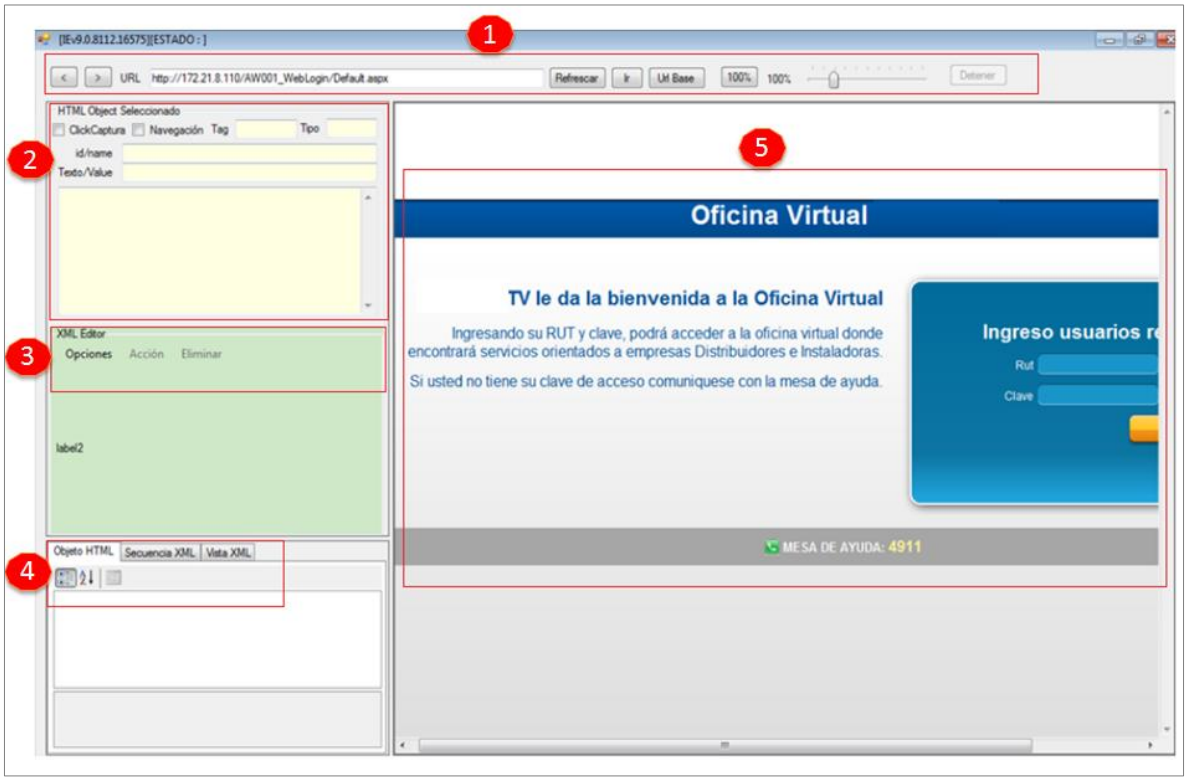

Imagen 31: GUI del módulo grabador de *scripts*.

A continuación se describen cada una de las partes que conforman la página del módulo grabador de *script*.

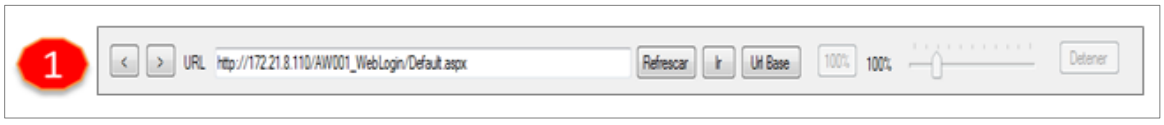

Imagen 32: Barra URL.

La barra de dirección permite capturar la página web de inicio que se requiere automatizar. En la imagen 32, se puede ver:

- Input: permite Ingresar dirección http de la aplicación que se desea grabar.
- Los botones permiten:
    - Refrescar: actualiza la página de la url cuando no se cargan algunas funcionalidades especiales producto de controles especiales que existen en el sistema, por ejemplo: los controles de Ajax.
    - Ir: cambiar la página de captura.
    - Url base: Utilizar la última url en la que se estuvo automatizando.
    - Detener: para dejar de grabar.

La línea de Zoom para aumentar o disminuir la visualización de la página visualizada en el punto 5.

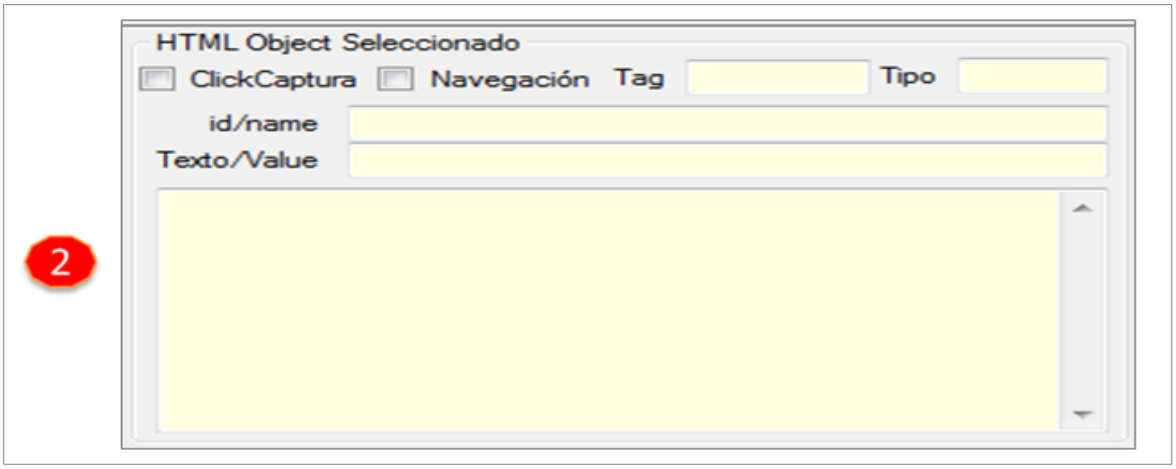

Imagen 33: HTML del objeto seleccionado.

Como se visualiza en la imagen 33, este objeto permite capturar y mostrar: el ID o nombre, valor y las propiedades del objeto seleccionado en la página que se navega en la sección 5.

La casilla ClickCaptura, al estar activada, permite obtener los datos del objeto seleccionado. Al dejar presionado el click del mouse por 2 segundos, obtiene el objeto relacionado de más alto nivel del objeto seleccionado. Si la casilla no se encuentra activada, solo capturará las propiedades del control que se visualiza.

La casilla Navegación, al estar activada, solo permitirá navegar por el sitio web, sin poder capturar los elementos de esta.

Imagen 34: Menú para crear XML de secuencias de captura.

Para crear o modificar un *script* automatizado se debe hacer click en Opciones del menú que se visualiza en la imagen 34. El cual permite crear un nuevo XML (estructura nueva de *script* de automatización), cargar un XML (*script* automatizado) y agregar nodos nuevos, los que pueden estar representados por Secuencias y Caso. En la imagen 35 se visualiza estas opciones.

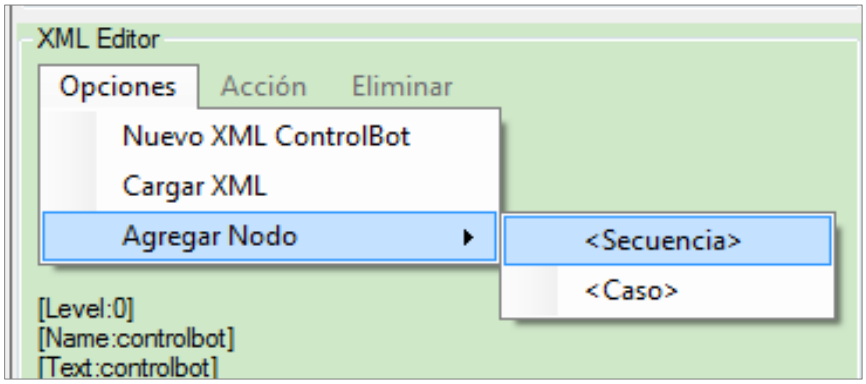

Imagen 35: Opciones del XML Editor.

La opción Secuencia corresponde a acciones en la GUI que pueden ser ejecutadas repetitivamente y no altera la funcionalidad en la aplicación. Por ejemplo: cuando se requiere llenar un formulario repetitivo como una encuesta de evaluación y que es requerido por el sistema para poder continuar.

La opción Caso permite realizar la automatización de los casos de prueba dinámicos, es decir, permite grabar las acciones en la GUI e indicar si el dato a comparar o ingresar provendrá desde un archivo externo. Por ejemplo: cuando se requiera automatizar *script* con datos variables para el completado de los inputs y que estos modifiquen el resultado de la prueba.

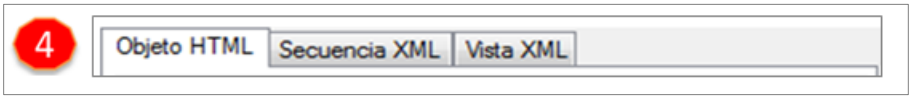

Imagen 36: Opciones de vista de los objetos.

Los botones que se visualizan en la imagen 36 permiten tener visibilidad en detalle del objeto HTML seleccionado, de la estructura del XML que se está automatizando y la vista global a detalle del XML automatizado.

Imagen 37: Browser de navegación.

La sección para navegación visualizada en la imagen 37, permite capturar las acciones a nivel de la GUI del sistema que se está automatizando, similar a lo que hace la herramienta Selenium IDE (capturar eventos de acuerdo a la acción realizada en una posición puntual dentro de la GUI), sin embargo, en complemento de las secciones 2, 3 y 4, permite la automatización basada en la captura de identificadores claves, por ejemplo: el id de los botones, caja de texto, títulos, entre otros, de tal forma que si la interfaz cambia no se requiere volver a automatizar, a menos que se modifiquen estos identificadores.

B.2. Módulo generador de datos de prueba

La interfaz de usuario que permite seleccionar los datos desde un archivo externo y asociarlos a los *scripts* automatizados se encuentra representada en la imagen 38, la cual, realiza el proceso en 3 pasos de acuerdo al menú de la imagen.

Imagen 38: GUI del módulo generador de datos de prueba.

El primer paso consiste en subir el archivo Excel que contiene los datos para las pruebas y posteriormente el archivo XML que contiene el *script* automatizado generado con el módulo grabador de secuencias el secuenciador. La imagen 39 representa la acción de subir archivo XML del *script* automatizado.

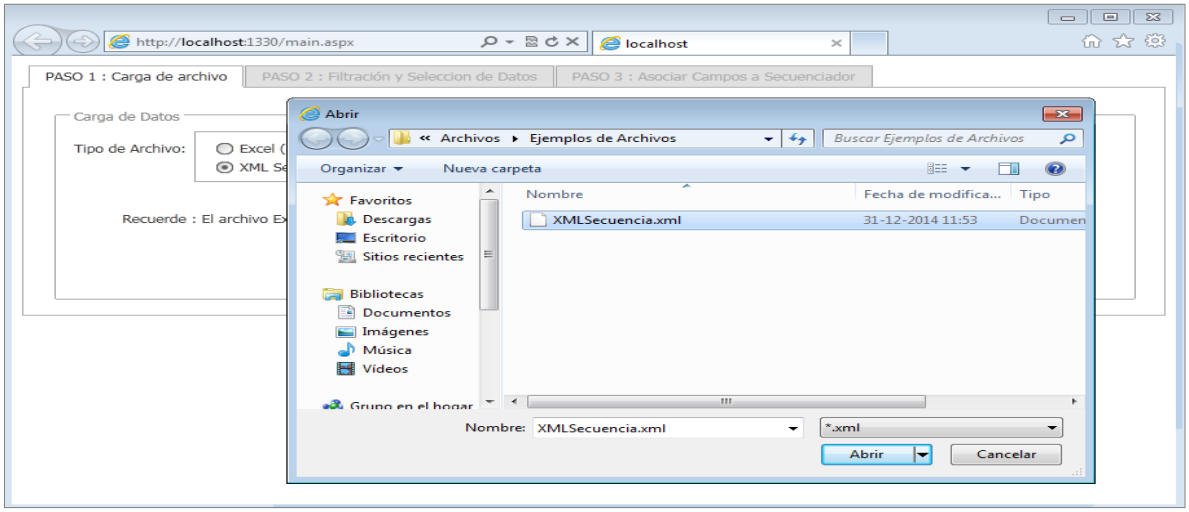

Imagen 39: Acción de subir archivo XML del *script* automatizado.

En el paso 2 se seleccionan los datos de prueba que se van a utilizar haciendo clic en las casillas. Además, se podrá filtrar los datos de todas las columnas que desee, con tan solo escribir las condiciones del filtro con operadores lógicos más complejos. La imagen 40 representa el paso 2 del proceso.

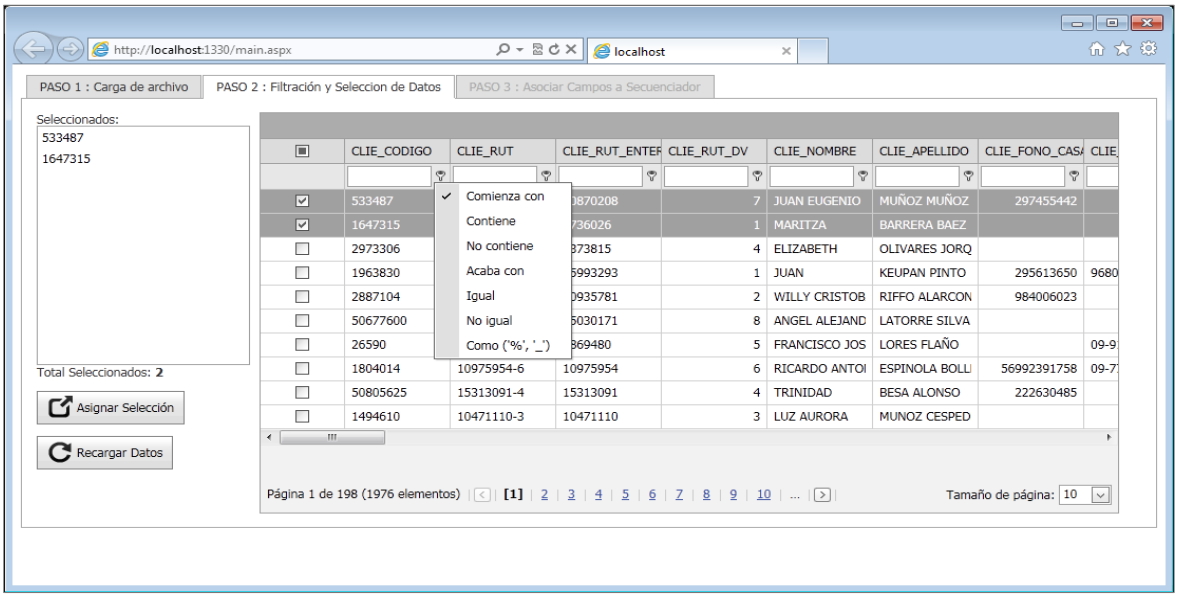

Imagen 40: Selección de datos.

Por último, en el paso 3 que se visualiza en la imagen 41, se puede observar en la parte superior izquierda de la imagen, el listado de los campos del archivo de datos previamente seleccionado y en la parte superior derecha de la imagen, la estructura tipo árbol del archivo XML del *script* automatizado.

En la parte inferior izquierda de la imagen se podrá ver los campos y nodos ya asociados. Y en la parte inferior derecha de la imagen mostrará el campo seleccionado y el nodo del *script* automatizado que será asociado una vez presionado el botón Asociar.

Para finalizar se debe hacer click en el botón Generar y Descargar Datos, el cual creará un nuevo archivo con el test de ejecución descrito en la sección 2.3 del proceso de transformación del *script.*

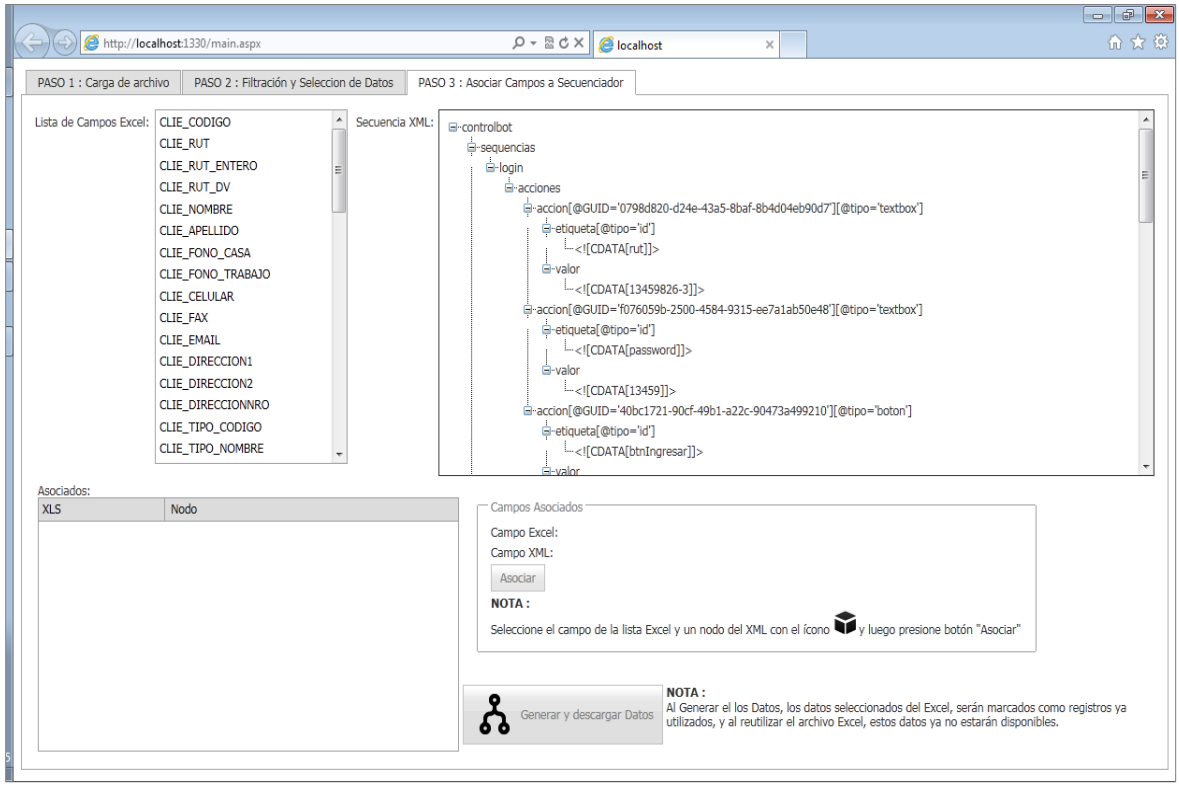

Imagen 41: Proceso de asociar campos a los *scripts*.