DESARROLLO PARA WEB SERVICE DE ESTRATEGIAS DE TOMA DE DECISIÓN CREDITICIA BASADO EN LA HERRAMIENTA DE DESARROLLO SM-POWERCURVE® STRATEGY MANAGEMENT

PROYECTO DE GRADO
MODALIDAD DE GRADO: PASANTÍA

PRESENTA:
CRISTIAN DAVID VARGAS MOSQUERA
CÓDIGO: 20122005012

DIRECTOR EXTERNO
INGENIERA PAOLA JAEL CASTRO JARAMILLO
GERENTE DE PROYECTO

DIRECTOR INTERNO
INGENIERO GUSTAVO ADOLFO PUERTO
DOCENTE FACULTAD DE INGENIERÍA
UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
FACULTAD DE INGENIERÍA
PROYECTO CURRICULAR INGENIERÍA ELECTRÓNICA
BOGOTÁ D.C
2018

Contenido

1. INTRODUCCIÓN

Por medio de **Desarrollo de software** se pueden generar diferentes soluciones que cumplan con alguna función específica que sea necesaria dentro de un proceso. Una empresa cuya actividad económica sea la venta de soluciones de software debe contar con diferentes medios para que los desarrollos puedan ser generados de una manera masiva y eficiente. La ingeniería de software dicta los términos que deben aplicarse en cada desarrollo realizado tales como etapas para aplicar en cada uno y ambientes disponibles para facilitar las labores en su creación.

La herramienta **SM – PowerCurve®** provee facilidades para generar soluciones de toma de decisión crediticia por medio de la utilización de repositorios compartidos y recursos como operaciones y funciones previamente establecidas. En el presente documento se describen los procesos que **Experian DataCredito™** emplea en el proyecto Decisor para ofrecer a sus usuarios una herramienta personalizada de toma de decisión crediticia consultada vía Web, utilizando la información que se encuentra en sus servidores obtenida a través de entidades fuentes de información

2. OBJETIVOS

2.1 OBJETIVO GENERAL

- Ejecutar el desarrollo de los requerimientos para las nuevas estrategias de toma de decisión en otorgamiento de crédito por medio del motor de desarrollo de lógica **SM (Power Curve® Strategy Managemet)**, realizar la configuración y generación de formularios en formato XML y catalogar en el **WEB SERVICE** ofrecido por **Experian DataCrédito™.**

2.2 OBJETIVOS ESPECÍFICOS

- Reconocer las etapas del desarrollo de software relacionado con plataformas web, evaluación crediticia y tratamiento de información financiera.

- Identificar la estructura de archivos basados en meta-lenguaje XML usado como estándar para el intercambio de información estructurada entre diferentes plataformas.

- Realizar el desarrollo de software de los requerimientos cumpliendo con las reglas de negocio y funcionalidades estrictamente especificadas en el documento de entendimiento.

- Reconocer los medios de obtención y tratamiento de la información semiprivada financiera y de otorgamiento de crédito.

- Gestionar el flujo de trabajo de requerimientos a través de la herramienta **IBM – Rational Clear Quest.**

3. MARCO TEÓRICO

3.1 DESARROLLO DE SOFTWARE A LA MEDIDA

Un alto porcentaje de empresas que se despliegan dentro del entorno productivo requieren utilizar herramientas de software que les permita controlar, planificar y ejecutar procesos de manera que estos sean cumplidos eficientemente. Teniendo en cuenta que cada empresa con el fin de crecer y desarrollarse genera un modelo de negocio único y diferenciado, es probable que herramientas de software estándar encontradas en el mercado no se adapten en su totalidad a los procesos de gestión que la empresa ya posee. En tal punto es donde emerge como solución el **Desarrollo de software a la medida** que es software específicamente desarrollado con los requerimientos que la empresa u organización solicite, este software parte desde las necesidades y, teniendo en cuenta las expectativas a futuro del solicitante, se adapta a sus procesos particulares.

3.2 FASES DEL DESARROLLO DE SOFTWARE

Por medio de la ingeniería de software se han definido fases dentro del desarrollo de soluciones con el fin de facilitar el proceso desde el inicio hacia sus posteriores etapas y además obtener mayor robustez en el producto final. A continuación se describen las etapas sugeridas para el desarrollo de soluciones de software [1].

3.2.1 Análisis de Requisitos

Se debe comprender en su totalidad el problema a resolver y las funcionalidades que el solicitante necesita que el sistema le provea. El objetivo es entonces especificar los requisitos funcionales y no funcionales, en esta fase del desarrollo se obtiene como resultado un documento de especificación de requerimientos del sistema y el cronograma del proyecto.

3.2.2 Diseño y arquitectura

Se refiere al establecimiento de las estructuras de datos, la arquitectura general del software, representaciones de interfaz y algoritmos [2]. Se definen los casos de uso y se transforman las organizaciones definidas en la etapa anterior en clases de diseño, obteniendo de esa manera un estándar cercano a la programación orientada a objetos.

3.2.3 Desarrollo

Durante esta actividad se traduce en una forma legible por la maquina el diseño generado en la etapa anterior del proceso. Se crean todos los componentes y módulos de software del sistema, dependiendo de los lenguajes de programación utilizados y los métodos de desarrollo empleados se determina la complejidad y duración de la etapa.

3.2.4 Pruebas

Consiste en comprobar el correcto funcionamiento de los procesos lógicos internos de cada componente que conforma el producto, se comparan los resultados obtenidos exclusivamente con la respuesta esperada estipulada en el documento de especificación de requerimiento generado en etapas previas. Se considera una buena práctica que las pruebas internas sean efectuadas por alguien diferente al desarrollador que programó el requerimiento, esta etapa también incluye pruebas realizadas por el solicitante del sistema.

3.2.5 Documentación y despliegue

Se genera todo tipo de documentación del desarrollo de software y gestión del proyecto, modelación, diagramas, pruebas, manuales de usuario, manual técnico, etc. [3] Esa documentación es entregada al cliente al mismo tiempo con la instalación de la solución de software generada en el servidor de producción del solicitante.

3.2.6 Mantenimiento

Generalmente es la fase más larga. El sistema es puesto en marcha y se realiza la corrección de errores descubiertos. Se realizan mejoras de implementación y se identifican nuevos requisitos. [4]

3.3 AMBIENTES DE DESARROLLO

Se le denomina **ambiente** al hardware y software donde se ejecuta una aplicación, en procesos de desarrollo se encuentra importante definir un espacio técnico donde los programadores puedan realizar pruebas del código generado sin que esto altere los servicios que el cliente posea en producción.

Durante el funcionamiento de una solución de software es normal que aún con el sistema ya instalado, algún tiempo después el cliente encuentre algún termino adicional que desee

agregarle o un detalle que no se visualizó durante el requerimiento anterior y necesita que se realice un ajuste. En ese punto es donde resulta importante tener un espacio con plena funcionalidad y copia de los binarios de la aplicación donde este ajuste pueda ser desarrollado y certificado paralelamente al funcionamiento del requerimiento anteriormente desplegado.

Usualmente se emplean diferentes servidores con interfaz idéntica (desarrollo, certificación, producción, etc.), todas con perfecta funcionalidad pero con información diferente en base de datos con el fin de permitir a los desarrolladores y al área de pruebas dar atención a los nuevos requerimientos sin tener interferencia con los productos ya alojados en el servidor de producción.

3.4 SM – POWER CURVE® STRATEGY MANAGEMENT

SM consiste en un ambiente de diseño de software, una plataforma de gestión de toma de decisiones desarrollada de carácter modular. Permite el desarrollo, monitoreo y diseño asistido de estrategias, el ambiente fue construido en Netbeans por tal razón diferentes estudios pueden ser desarrollados para soluciones específicas de negocio y puede ser configurado para diferentes tipos de usuarios asignando diferentes roles y permisos de modificación.

Todos los componentes generados en los desarrollos son guardados en repositorios dentro de los cuales los archivos pueden ser usados para gestionar componentes. Plantillas completas o módulos pueden ser reutilizados como parte de una nueva solución que este siendo desarrollada. La habilidad de controlar plantillas dentro de un repositorio común permite la fácil aplicación en desarrollos estándar y también facilita la reutilización de componentes para especializar algunas piezas como sea requerido.

Los flujos de decisión son los elementos clave de control y permiten agrupar una serie de tareas para ejercer una decisión. Estos flujos pueden hallarse embebidos en un flujo de trabajo como un nodo y la decisión termina construida implementando herramientas como análisis de puntajes, arboles de decisión, ramas condicionales y flujos enlazados dentro de una secuencia.

3.4.1 Scripts

DERIVED DATA SCRIPTS

Se trata de un archivo de ordenes cuyo contenido es un archivo de texto plano, el código programado por el desarrollador donde se ejecutan las sentencias, los procesos y condicionales utilizando los procesos que permite utilizar la paleta de recursos del software (Véase Anexo).

DERIVED DATA SCRIPTS TREE

Consiste en una interfaz creada para relacionar hasta tres archivos de texto **Derived Data Scripts,** de tal manera que la maquina los ejecute uno tras otro de manera secuencial conforme al orden que el programador ubique los archivos dentro de la pila.

USER DEFINED FUNCTIONS

Es un conjunto de declaraciones en código con el fin de realizar una tarea específica y retornar algún valor. Utilizando los parámetros enviados cuando se hace el llamado de la función ejecuta de manera secuencial las sentencias encontradas en el código y según las condiciones presentadas envía un valor de retorno.

INTERACTIVE TEST PARA SCRIPT

Se trata de una herramienta capaz de compilar el código desarrollado dentro de un Script en particular. Se le permite al desarrollador asignar los valores que requiera para cada una de las variables de entrada utilizadas dentro del script y observar cual es el comportamiento de salida del Script según los parámetros introducidos, útil para realizar pruebas unitarias y verificar el funcionamiento de cada sección de código.

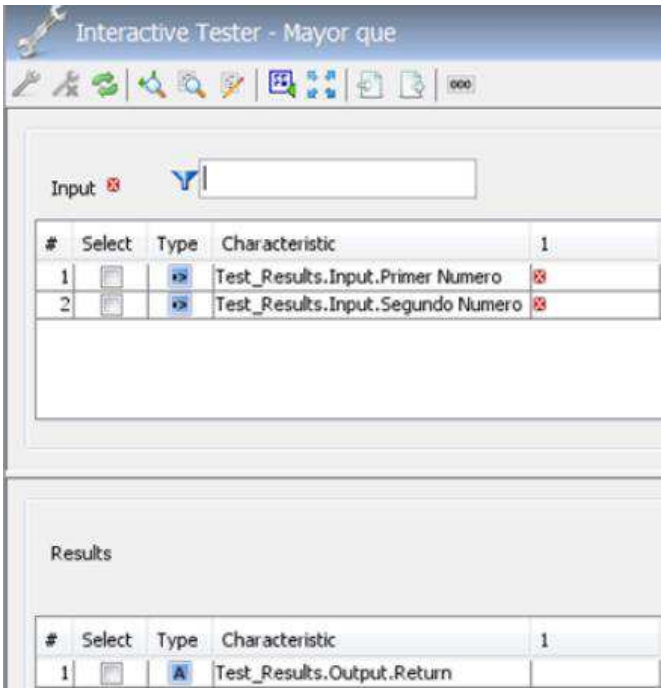

Figura 1 Interactive Test. Recuperada de: Experian (2015), Strategy Design Studio.

3.4.2 Proceso de Deploy

CHECK – IN

Con el objetivo de actualizar todos los componentes al repositorio compartido [5], se efectúa cuando el desarrollo se encuentra completo y listo para desplegar la solución, actualiza a todos los demás usuarios los nuevos componentes desarrollados.

ASIGNACIÓN DE ETIQUETA DE FLUJO

Fusionar un flujo de proceso de decisión junto con una call interface en una etiqueta única, de manera que permita realizarse el despliegue de la solución [5].

DESPLIEGUE

Ejecuta la etiqueta generada con el fin de generar los archivos que el agente de decisión requiere para implementar la solución en los ambientes de desarrollo.

3.5 IBM® RATIONAL CLEARQUEST

Diseñado para ayudar a gestionar el ciclo de vida de soluciones de software de una manera más efectiva, provee acceso a la información necesaria con el fin de tomar mejores decisiones. Consiste en una herramienta que permite ser consultada vía Web donde se alojan todos los requerimientos de desarrollo de software que tiene una compañía, brindando la posibilidad de realizar consultas sobre el estado del desarrollo de cada requerimiento en particular, lo anterior gracias a que a cada solicitud se le es asignado un número único de identificación que posteriormente será utilizado para gestionar las fases del desarrollo.

Las ventajas de utilizar IBM® RATIONAL CLEARQUEST como herramienta de gestión de desarrollo de software son las siguientes [6]:

- Mejoramiento en la visibilidad y el control: Los sistemas dispares podrían crear problemas en su coordinación, la herramienta permite personalizar los procesos de desarrollo y tener una vista integrada de todo el proyecto.

- Colaboración en equipo: Optimiza la integración del equipo integrando elementos que usualmente son aislados, generando productividad utilizando flujos de trabajo automatizados y notificaciones vía email alertando a los miembros cuando su trabajo sea requerido.

- Aumento en la calidad del software: Ayuda a eliminar errores de software que aparecen cuando se ejecutan procesos manuales o pérdidas de comunicación o informaciones erradas.

- Habilitando desarrollos seguros: Puede aplicarse autenticaciones por usuario los cuales permiten firmas electrónicas y colocan huellas electrónicas con las cuales sea sencillo realizar procesos de auditoria en caso de requerirlo.

3.6 WEB SERVICE DESCRIPTION LANGUAJE (WSDL)

Debido a que los protocolos de comunicación y formatos de mensaje están estandarizados en la comunidad web se hace necesario tener la capacidad de describir las comunicaciones de alguna manera estructurada. WSDL da solución a esa necesidad definiendo un vocabulario de **Extensible Markup Language (XML)** para describir servicios en la red como un conjunto de puntos finales con la posibilidad de intercambiar mensajes.

Un **WSDL** consiste en un documento **XML** que define requisitos, parámetros, operaciones y respuestas empleadas en la interacción vía servicio web. Define un mecanismo estándar para realizar transmisión de información por vía de mensajes **XML,** haciendo posible acceder un recurso solamente transmitiendo estos mensajes a través de protocolos estándar como TCP. HTTP o SMTP reduciendo el nivel de complejidad para los potenciales consumidores.

Los elementos que componen un WSDL son:

ELEMENT NAME	DESCRIPTION
types	Un contenedor para definiciones de tipo abstracto determinada por el esquema XML
message	Una definición de un mensaje abstracto, que puede consistir de múltiples partes, cada parte puede ser de un tipo diferente.
portType	Un conjunto abstracto de operaciones suportadas por uno o más puntos finales; las operaciones son definidas por un intercambio de mensajes.

binding	Un protocolo concreto y especificación del formato de la información para un **portType** particular.
service	Una colección de puntos finales relacionados, donde un punto final está definido como la combinación de un **binding** y una dirección.

Fuente: World Wide Web Consortium[7]

3.6.1 Types

Las definiciones de **Type** ubicadas en este espacio son referenciadas por definiciones de mensaje de un más alto nivel con el objetivo de precisar los detalles de estructura del mensaje.

```
<definitions .... >
    <types>
        <xsd:schema .... />*
    </types>
</definitions>
```

Figura 2: Estructura básica de definición de un elemento **Type**

3.6.2 Messages

Especifica un mensaje abstracto que puede usarse como entrada o salida de una operación. Consiste en una o varias partes de elementos donde cada parte es asociada con un elemento o un tipo.

```
<definitions .... >
    <message name="nmtoken"> *
        <part name="nmtoken" element="qname"? type="qname"?/> *
    </message>
</definitions>
```

Figura 3: Estructura básica de definición de un elemento **Message**.

3.6.3 PortTypes

Declara las operaciones que pueden ser realizadas y los mensajes que se encuentran involucrados.

```
<definitions .... >
    <portType name="nmtoken">
        <operation name="nmtoken" .... /> *
    </portType>
</definitions>
```

Figura 4: Estructura básica de definición de un elemento **PortType**

3.6.4 Binding

Describe los detalles concretos del uso de un **PortType** particular con un protocolo dado. El Binding contiene tantos elementos de extensión como operaciones en el **PortType** que este describe.

```
<wsdl:definitions .... >
    <wsdl:binding name="nmtoken" type="qname"> *
        <-- extensibility element providing binding details --> *
        <wsdl:operation name="nmtoken"> *
           <-- extensibility element for operation details --> *
           <wsdl:input name="nmtoken"? > ?
               <-- extensibility element for body details -->
           </wsdl:input>
           <wsdl:output name="nmtoken"? > ?
               <-- extensibility element for body details -->
           </wsdl:output>
           <wsdl:fault name="nmtoken"> *
               <-- extensibility element for body details -->
           </wsdl:fault>
        </wsdl:operation>
    </wsdl:binding>
</wsdl:definitions>
```

Figura 5: Estructura básica de un elemento **Binding**

3.6.5 Service

Define un grupo de puntos finales que exponen un elemento particular **Binding.**

```
<definitions .... >
    <service .... > *
        <port name="nmtoken" binding="qname"> *
           <-- extensibility element defines address details -->
        </port>
    </service>
</definitions>
```

Figura 6: Estructura básica de un elemento **Service**

3.7 ENTORNO DE OBTENCIÓN Y MANIPULACIÓN DE LA INFORMACIÓN PERSONAL

Experian DataCredito™ es una entidad que gestiona información personal crediticia, financiera, comercial y de servicios de muchos titulares, teniendo en cuenta que se trata de información sensible, la entidad debe regirse y certificar el cumplimiento de leyes gubernamentales que se han implementado con el fin de establecer una metodología clara sobre el correcto proceder en el mantenimiento de este tipo de datos.

Las leyes de tratamiento de datos que son aplicables a Experian DataCredito™ son las siguientes: Ley estatutaria 1581 de 2012 donde se dictan disposiciones generales para la protección de datos personales[8] y la ley 1266 de 2008 Hábeas Data que fue limitada exclusivamente a regular el denominado hábeas data financiero, donde se describe el derecho constitucional fundamental que permite a los ciudadanos conocer y rectificar información financiera que se haya recogido sobre ellos en bancos de datos[9].

3.7.1 Tipos de perfiles de tratamiento de datos

La ley 1266 de 2008 de Hábeas Data define con precisión cuales son los perfiles que están autorizados para gestionar información únicamente financiera de los usuarios; estos perfiles son [8]:

- **Titular de la información**: La persona natural o jurídica a quien se refiere la información que reposa en un banco de datos.

- **Fuentes de información**: Las entidades, empresas o personas con las cuales el titular ha entablado una relación financiera o comercial y le entregó información autorizando expresamente el reporte de sus datos sobre el cumplimiento o incumplimiento de sus obligaciones a una central de riesgos. Son estas fuentes de información quienes suministran la información de los titulares a los operadores.

- **Operadores de la información**: Reciben la información personal de carácter financiero, crediticio, comercial y de servicios que envían las fuentes y que previamente ha sido recabada de los titulares. A estos operadores, como por ejemplo las centrales de riesgo (Experian Datacredito™), reciben una gran cantidad de datos de las fuentes para compartirla con sus usuarios en los términos que dictamina la ley

- **Usuarios de información**: Entidades financieras, crediticias, comerciales y de servicios ante las cuales el titular solicita un producto o servicio y requieres consultar la historia crediticia del mismo. Pueden utilizar la información como elemento de análisis de capacidad de endeudamiento con el fin de establecer una relación contractual.

3.8 SEGURIDAD DE LA INFORMACIÓN

Teniendo en cuenta que la información que gestiona Experian Datacredito™ es de carácter semiprivado y que se almacenan datos crediticios y financieros de millones de usuarios, se dispone de unas estrictas normas de limitación de acceso al medio y tratamiento de la información.

3.8.1 Red Virtual Privada

Una VPN, por sus siglas en inglés (Virtual Private Network) consiste en un modelo de red capaz de generar una semejanza a un túnel donde los paquetes de datos de una red privada viajan por un medio definido dentro de la red pública. Considerando que se trata de un proyecto de soporte inter-empresarial cuyos espacios físicos se encontraban distantes se proveyó de un medio para un intercambio bidireccional de información seguro.

El software de red privada virtual que se dispuso para la ejecución provee un medio de comunicación seguro y cifrado de extremo a extremo que utiliza protocolos SSL (Secure Socket Layer). Se trata de protocolos criptográficos que ofrecen privacidad e integridad de la información entre dos extremos, lo cual asegura que la información que circula sobre la red no podrá ser interceptada o modificada durante el tránsito por entes no autorizados. Siguiendo el modelo de arquitectura de redes por capas (OSI), el protocolo SSL es implementado entre la capa de aplicación y la capa de transporte.

La red privada virtual posee dos tipos de funcionamiento:

- **TUN:** Implementado para generar túneles virtuales que operan con el protocolo IP, encapsulando de esta manera todos los paquetes que se envíen por medio de él cómo datagramas TCP o UDP aunque las maquinas en cada extremo pertenecerán a subredes diferentes
- **TAP:** Utilizado para enlazar usuarios remotos, encapsulan directamente los paquetes Ethernet que pueden ser diferente a IP permitiendo que a las máquinas detrás de los extremos les sea permitido funcionar como parte de la misma red.

Además de proveer un medio de comunicación confidencial, íntegro y disponible, al aplicar la VPN en el modo de funcionamiento TAP, se realiza la asignación de la dirección IP de salida dentro de un grupo de direcciones previamente diseñadas que al ser configuradas dentro de las normas de firewall en el servidor permitirá el libre tráfico de información cifrada entre los dos extremos remotos.

Otro beneficio al emplear la conexión remota a través de una red privada virtual es que el administrador de la red emite a cada usuario un certificado único de seguridad, que involucra un nombre de usuario y contraseña de acceso que puede ser revocado cuando el administrador lo crea pertinente.

A continuación se evidencia la diferencia en la respuesta a la solicitud de comunicación hacia una dirección IP que solo es accesible en la red interna del servidor:

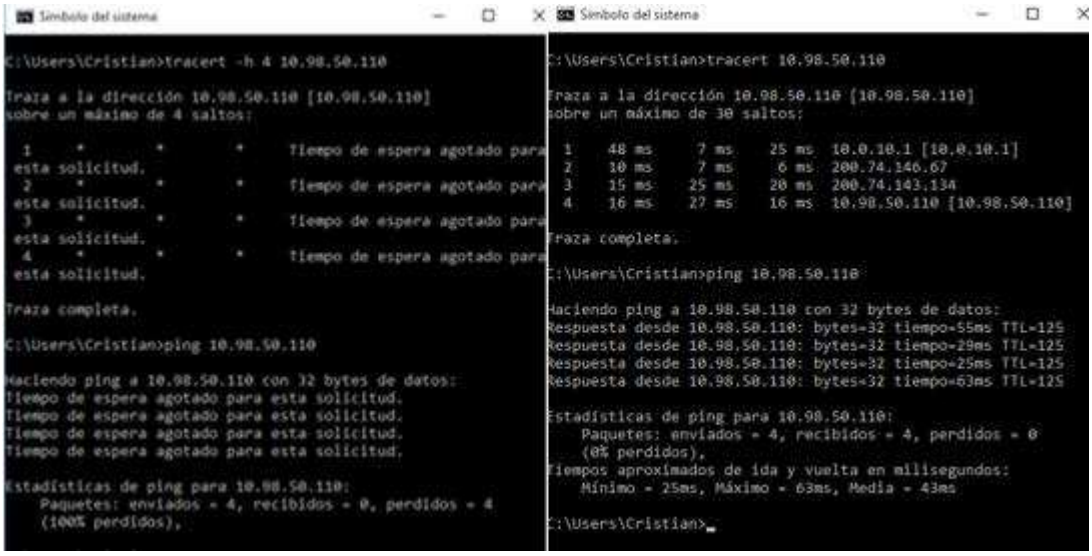

Figura 7: Respuesta de solicitud de comunicación en el servidor.

Como se puede apreciar en la parte izquierda de la imagen al intentar trazar la ruta de conexión y comprobar comunicación teniendo como único medio la Internet no se obtiene respuesta del servidor, en la parte derecha de la imagen se evidencia que si se obtiene respuesta de comunicación y existe una ruta de conexión toda vez que la VPN ha sido activada y asigna una dirección IP de salida que fue configurada con permisos de acceso en las normas de firewall del servidor.

4. DESARROLLO DEL PLAN DE TRABAJO

Durante la planificación del cronograma de actividades planteado en el anteproyecto del presente trabajo se estipularon etapas que debían ser llevadas a cabo, a continuación se describen los objetivos cumplidos en cada fase:

4.1 FASE 1: ESTUDIO Y CAPACITACIÓN EN LA HERRAMIENTA DE DESARROLLO SM – POWER CURVE® STRATEGY MANAGEMENT Y DEMAS AMBIENTES DE DESARROLLO

Durante esta fase, se realizó un estudio de la herramienta **SM – PowerCurve** con todos sus componentes, utilizando el contenido de ayuda propio de la herramienta y un documento de AutoTraining que es entregado junto con la instalación del producto en el equipo.

4.2 FASE 2: ANÁLISIS DE LA ESTRUCTURA DE LOS REQUERIMIENTOS

Teniendo en cuenta que los servicios ofrecidos consisten en desarrollos personalizados para cada entidad según los casos que este requiera, se ha implementado un formato de documento de descripción del requerimiento para lograr abarcar todos los alcances que el cliente necesite. Se debió realizar un estudio minucioso sobre esa estructura general de los documentos para comprender en su totalidad cada aspecto que los clientes resaltan que requiere su solución.

4.3 FASE 3: DESARROLLO DE LAS ESTRATEGIAS DE TOMA DE DECISIÓN

Una vez realizada la conceptualización con las herramientas de desarrollo y los documentos de descripción se llevó a cabo el desarrollo completo de las nuevas estrategias generando formularios, desarrollando la lógica necesaria para cumplir con las reglas de negocio. Por medio del motor SM–PowerCurve generando artefactos para desplegarlos sobre el WEB SERVICE y finalmente realizando las debidas pruebas unitarias de funcionamiento en los ambientes de desarrollo de pruebas internas y de muestra.

4.4 FASE 4: DOCUMENTACION Y SOCIALIZACION DE RESULTADOS

La fase de documentación fue realizada al mismo tiempo con cada una de las etapas de avance del presente proyecto, en este documento se describen los procedimientos para el correcto desarrollo de los objetivos planteados, los conceptos sobre aspectos y herramientas aprendidos para obtener los resultados esperados.

5. DESARROLLO DEL PROYECTO

El núcleo del proyecto encuentra sustento en el modelo creado por Experian DataCrédito™ de un área de negocio que provee un sistema de consulta WEB a diferentes entidades que ofrecen otorgamiento de crédito. Esta herramienta es capaz de determinar con agilidad y efectividad la viabilidad de otorgar un crédito a una persona natural o jurídica según su historial de vida crediticia siguiendo las normas de otorgamiento propias de cada entidad, esta solución de desarrollo para toma de decisión de otorgamiento crediticio es llamada ESTRATEGIA.

A cada necesidad de solución se le asigna un número único de identificación de requerimiento que se emplea para gestionar cada solución a través de la herramienta de IBM® Rational ClearQuest además de adjuntar a cada uno su respectivo documento de especificación de requerimiento.

5.1 ESTRUCTURA DEL DOCUMENTO DE ESPECIFICACIÓN DE REQUERIMIENTOS

- IDENTIFICACIÓN DE ENTIDAD Y ESTRATEGIA: Este espacio es empleado para identificar la entidad para la cual se está desarrollando la solución, cual es el nombre de identificación de la estrategia, el número de identificación de la misma y la complejidad del desarrollo. Es importante resaltar que tanto el nombre como la identificación de cada estrategia es único e irrepetible.

- DEFINICIÓN DEL FLUJO: Se realiza aclaración de la manera cómo funciona la estrategia en particular, puede ser de flujo lineal o cascada. En el flujo lineal la salida web muestra todas las causales por las cuales se obtuvo determinada decisión y en el flujo cascada se muestra junto con la decisión la primera causal por la cual se toma tal decisión.

- DEFINICIÓN DE REGLAS DE NEGOCIO: Estas reglas hacen referencia a las normas primarias que debe cumplir la estrategia, como cuales decisiones puede proyectar en la salida y las restricciones que rigen la forma en que debe operar la aplicación.

- DESCRIPCIÓN ALCANCE Y REQUERIMIENTOS FUNCIONALES: En este apartado se especifica toda la lógica que la estrategia debe cumplir, donde se describen a profundidad las razones por las cuales se toma determinada decisión según el hábito crediticio de la persona que está siendo consultada. Es importante resaltar que en este apartado se describen los campos de información requeridos en los formularios de entrada y de igual manera los campos que se deben mostrar como variables de salida.

5.2 DESARROLLO DEL REQUERIMIENTO

Experian DataCrédito™ provee tres ambientes de desarrollo diferentes, cuya interfaz gráfica es exactamente igual pero la información en las bases de datos de cada ambiente no es la misma, esto con el fin de permitir el libre desarrollo de las soluciones de software sin interrumpir los productos anteriormente desplegado en producción y tampoco obstaculizar la realización de la etapa de pruebas a los requerimientos desarrollados.

Para el desarrollo de cada requerimiento se realiza la configuración de los formularios de entrada y salida de manera estrictamente igual a como se visualiza plasmado en el documento de especificación de requerimientos. Dentro del ambiente de desarrollo se asocian las variables de entrada requeridas así como las variables de salida, como se muestra a continuación:

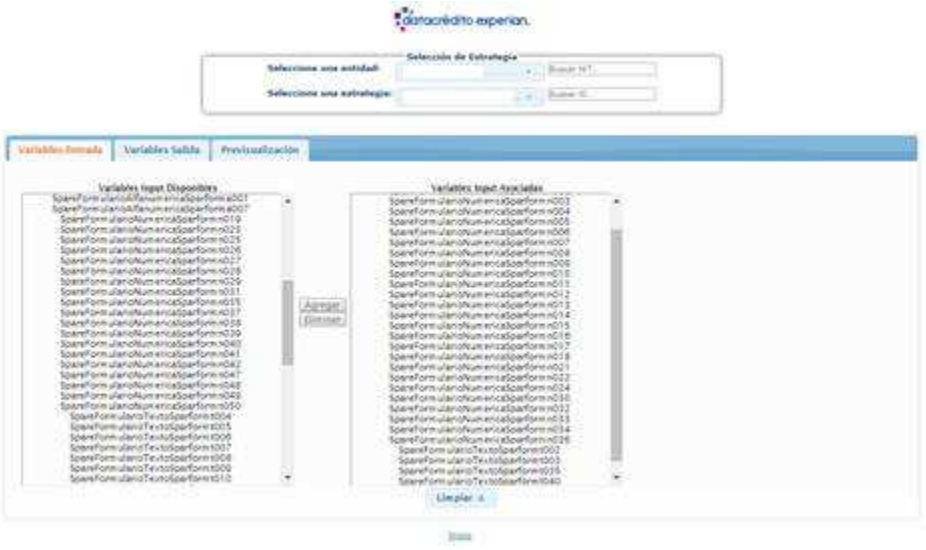

Figura 8: Configuración de variables de entrada por formulario

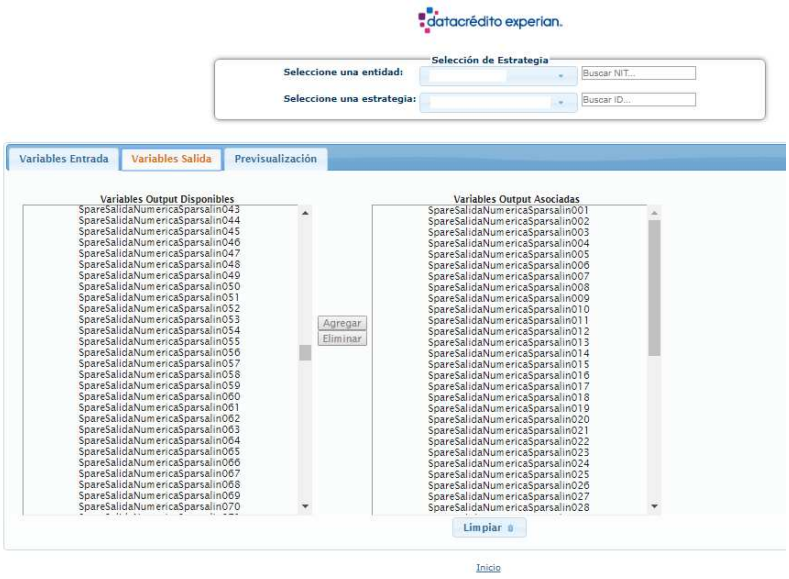

Figura 9: Configuración variables de salida Web

Concluida la configuración completa de los formularios en el ambiente de desarrollo se podrán generar los archivos correspondientes a la estrategia en formato XML, cuyo contenido es principalmente la descripción de la información que se requiere que quien realiza la consulta conceda diligenciando el formulario y la solicitud de información que se requiere de la base de datos del servidor.

Los archivos XML de una estrategia son dos:

- Formulario: Describe en líneas básicas los aspectos gráficos de interfaz que tiene el formulario de entrada, los campos de entrada, el nombre que tiene cada campo de entrada y detalla las opciones de selección para esos campos de tipo lista desplegable.

```
<Request><Invocation><StrategyName/></Invocation><Form><Field><Type>string</Type><Code>Edad</Code><PromptText>Edad</PromptText><RenderingInformation><RenderAs>
TextField</RenderAs><MaxLength>20</MaxLength></RenderingInformation></Field><Field><Type>string</Type><Code>Tiempo_Actividad</Code><PromptText>Tiempo Actividad
</PromptText><RenderingInformation><RenderAs>TextField</RenderAs><MaxLength>20</MaxLength></RenderingInformation></Field><Field><Type>string</Type><Code>
Ingresos_Fijos</Code><PromptText>Ingresos Fijos</PromptText><RenderingInformation><RenderAs>TextField</RenderAs><MaxLength>20
</MaxLength></RenderingInformation></Field><Field><Type>string</Type><Code>Ingreso_Variable</Code><PromptText>Ingreso Variable
</PromptText><RenderingInformation><RenderAs>TextField</RenderAs><MaxLength>20</MaxLength></RenderingInformation></Field><Field><Type>string</Type><Code>
Ciudad_Zona</Code><PromptText>Ciudad Zona</PromptText><RenderingInformation><RenderAs>DropDown</RenderAs><MaxLength>30</MaxLength><ValueListItem><OptionText>1
</OptionText><OptionValue>Zona Sur</OptionValue></ValueListItem><ValueListItem><OptionText>2</OptionText><OptionValue>Zona Centro
</OptionValue></ValueListItem><ValueListItem><OptionText>3</OptionText><OptionValue>Zona Oriente</OptionValue></ValueListItem><ValueListItem><OptionText>4
</OptionText><OptionValue>Zona Occidente</OptionValue></ValueListItem></RenderingInformation></Field><Field><Type>string</Type><Code>Actividad
</Code><PromptText>Actividad</PromptText><RenderingInformation><RenderAs>DropDown</RenderAs><MaxLength>500</MaxLength><ValueListItem><OptionText>1
</OptionText><OptionValue>Empleado</OptionValue></ValueListItem><ValueListItem><OptionText>2</OptionText><OptionValue>Empleado Informal
</OptionValue></ValueListItem><ValueListItem><OptionText>3</OptionText><OptionValue>Independiente
</OptionValue></ValueListItem></RenderingInformation></Field></Form><Decisons/></Request>
```

Figura 10: Muestra de un archivo de formulario en lenguaje XML

- Formulario Extendido: Contiene el detalle de todos los campos que configuran el formulario además del request con la información solicitada del servidor. Para los campos de escritura muestra los límites de la longitud en cantidad caracteres y del número en valor del dato y para los campos de listas de selección describe el valor que se muestra al usuario y el valor que envía a la estrategia para cada opción.


```
<?xml version="1.0" encoding="UTF-8"?> <ConfiguracionFormulario generarPreguntasAleatorias="false" variablesDisponibles=""><ValidacionesCampo
isRespuestasAleatorias="false" numeroMinimo="0.0" numeroMaximo="0.0" longitudMinima="0" longitudMaxima="0" code="Actividad" orden="2"><ListaValores optionText
="1" optionValue="Empleado" /><ListaValores optionText="2" optionValue="Empleado Informal" /><ListaValores optionText="3" optionValue="Independiente" />
</ValidacionesCampo><ValidacionesCampo isRespuestasAleatorias="false" numeroMinimo="0.0" numeroMaximo="0.0" longitudMinima="0" longitudMaxima="0" code="Edad"
orden="1" /><ValidacionesCampo isRespuestasAleatorias="false" numeroMinimo="0.0" numeroMaximo="0.0" longitudMinima="0" longitudMaxima="0" code=
"Tiempo_Actividad" orden="3" /><ValidacionesCampo isRespuestasAleatorias="false" numeroMinimo="0.0" numeroMaximo="0.0" longitudMinima="0" longitudMaxima="0"
code="Ingresos_Fijos" orden="4" /><ValidacionesCampo isRespuestasAleatorias="false" numeroMinimo="0.0" numeroMaximo="0.0" longitudMinima="0" longitudMaxima=
"0" code="Ingreso_Variable" orden="5" /><ValidacionesCampo isRespuestasAleatorias="false" numeroMinimo="0.0" numeroMaximo="0.0" longitudMinima="0"
longitudMaxima="0" code="Ciudad_Zona" orden="6"><ListaValores optionText="1" optionValue="Zona Sur" /><ListaValores optionText="2" optionValue="Zona Centro"
/><ListaValores optionText="3" optionValue="Zona Oriente" /><ListaValores optionText="4" optionValue="Zona Occidente" /></ValidacionesCampo><StrategyGSon>
```


Figura 11: Muestra de un archivo de formulario extendido en lenguaje XML

Posteriormente se despliegan en la herramienta de desarrollo SM – PowerCurve® Strategy Management utilizando el lenguaje máquina que este requiere, los requerimientos funcionales según se encuentren descritos, creando variables locales, asignando valores y empleando las sentencias lógicas que sean necesarias.

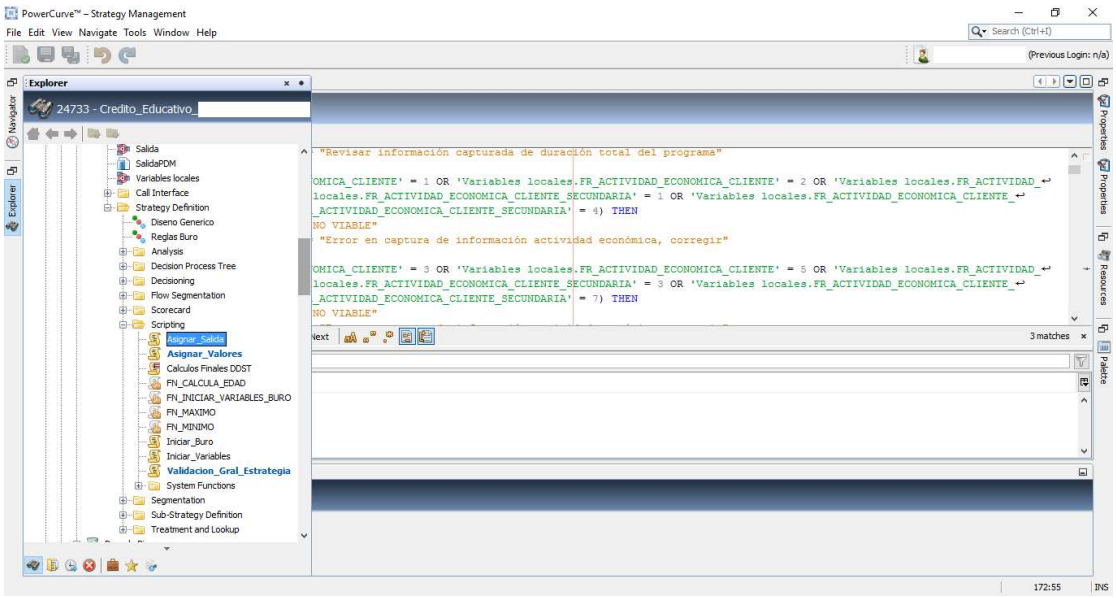

Figura 12: Herramienta de desarrollo SM - PowerCurve

Una vez culminado el desarrollo lógico dentro de la herramienta de desarrollo se procede a realizar el proceso de Deploy de la estrategia generada, este proceso finaliza con la generación de ciertos artefactos que posteriormente son desplegados en el ambiente de desarrollo pertinente al caso. Una vez llevados a cabo todos los procesos de configuración, desarrollo y despliegue mencionados anteriormente la estrategia es apta para realizar pruebas unitarias con el fin de verificar funcionamiento.

5.3 PRUEBAS UNITARIAS

El ambiente permite verificar el funcionamiento de estrategias ejecutando pruebas cuyos datos obtenidos se encontraran asociados a la información encontrada en las bases de datos articuladas al ambiente de pruebas.

El procedimiento para realizar una prueba unitaria es en líneas generales el siguiente:

- Se selecciona la estrategia que se desea consultar vía web lo cual mostrara los formularios de entrada asociados a la estrategia:

Figura 13: Procedimiento consulta en plataforma web

- Se diligencia la información del formulario para el usuario que va a ser consultado por el servicio web y posteriormente se procede a consultar la estrategia, obteniendo los resultados respectivos con los cálculos desarrollados dentro de la herramienta de software y las variables de salida asociadas para esa estrategia, en particular del caso de la persona consultada como se muestra a continuación:

Estudio

Información Básica	
Nombres y Apellidos:	
Tipo de Documento:	CC
Número de Documento:	
Edad:	29 a 35 años
Género:	Masculino
Resultados y variables	
Fecha consulta:	2018-05-29 11:08:10
Decisión:	ESTUDIO
Puntaje:	0.0
Score:	- Acierta +=753
Respuesta personalizada:	- Decision=ESTUDIO - Causal=Huellas de consulta - Ingresos_Netos=1510943.02000 - Total_Cuotas=539056.98000 - Ingresos_Variables_Calculados=100000 - Gastos_Familiares=1050000.00 - Porcentaje_De_Castigo=0 - Num_Cartera_Castigada=0 - Num_Dudoso_Recaudo=0 - Num_Mora30_Vigente=0 - Num_MoraMayor60_Vig=0 - Num_MorHist30_RC=0 - Num_MorHist30_OTP=0 - Num_MoraHist_60=0 - Num_MorHist_90=0 - Calificacion=0 - Aperturas=0 - Huellas=98

Condiciones de uso | Mapa del sitio

Figura 14: Respuesta personalizada de la estrategia

Finalmente, comprobado el correcto funcionamiento de la estrategia desarrollada todos los artefactos generados son homologados en el ambiente dispuesto para el área de calidad de desarrollo quienes realizan las evaluaciones pertinentes del caso.

5.4 GESTIÓN DEL REQUERIMIENTO

Se dispone de la herramienta IBM® Rational ClearQuest para la gestión de desarrollo de software, por medio de este software se es asignado a cada requerimiento un número único de identificación y el software posee estados definidos donde se puede comprobar el flujo de trabajo que ha tenido el requerimiento. Los estados asociados a los requerimientos de desarrollo de estrategias son descritos a continuación:

ESTADO	DESCRIPCIÓN
CREADA	Se genera el requerimiento bajo previas reuniones de entendimiento y generación del documento de especificación de requerimiento adjunto.
PREPARACION AMBIENTE	En esta etapa se deben crear las nuevas soluciones a desarrollar tanto en los diferentes ambientes como en la herramienta de desarrollo **SM**
REQUERIMIENTO RECIBIDO	La solicitud permanece en este estado mientras se es asignado el requerimiento a un líder técnico
PLANEACIÓN	La solicitud permanece en este estado hasta asignar el requerimiento a su respectivo desarrollador
CONSTRUCCIÓN	En este estado permanece el requerimiento mientras es configurada y desarrollada toda la lógica que el mismo solicita.
ENTREGA PRUEBAS	Se cargan los todos los artefactos asociados a la solución de software y se realizan pruebas unitarias.
DESPLEGAR AMBIENTE PRUEBAS	Se diligencian y se adjuntan documentos que certifican realización de pruebas unitarias y su correcto funcionamiento
ENTREGAR A AMBIENTE PRUEBAS	Se despliega la solución en el ambiente utilizado por pruebas y se envía a realizar pruebas de parte del área de calidad de desarrollo.

CERTIFICACIÓN CLIENTE	Las pruebas en el área de calidad fueron satisfactorias y la solución se adecua para pruebas del cliente.
POSTINSTALACIÓN	Es instalada la solución de software desarrollada en los servidores del cliente

Vale la pena resaltar sobre el estado de ENTREGAR A AMBIENTE PRUEBAS que puesto que en este periodo el área de calidad envía pruebas por lotes de consultas para hacer la respectiva certificación del código de desarrollo, es probable que algún requerimiento funcional no esté funcionando de la manera adecuada dentro de la solución de software. De suceder así el área de pruebas retrocede el requerimiento al estado ENTREGA PRUEBAS donde el desarrollador verifica los hallazgos reportados y de ser necesario modificar los artefactos este debe retroceder el requerimiento al estado CONSTRUCCIÓN para poder realizar la tarea.

Otro aspecto importante sobre la gestión del requerimiento es que la herramienta utilizada cada vez que se necesite modificar artefactos de una solución requiere añadir una observación de acuerdo al motivo de actualización. Esto permite generar reportes sobre la cantidad de hallazgos en pruebas de calidad y evaluar permanentemente el desempeño de los desarrolladores con respecto a sus soluciones de software.

Teniendo en cuenta la explicación anterior sobre los estados dentro de los cuales se puede encontrar un requerimiento, se puede representar de una manera gráfica, por medio de un diagrama de flujo, el ciclo de vida de cada requerimiento. La representación seria de la siguiente manera:

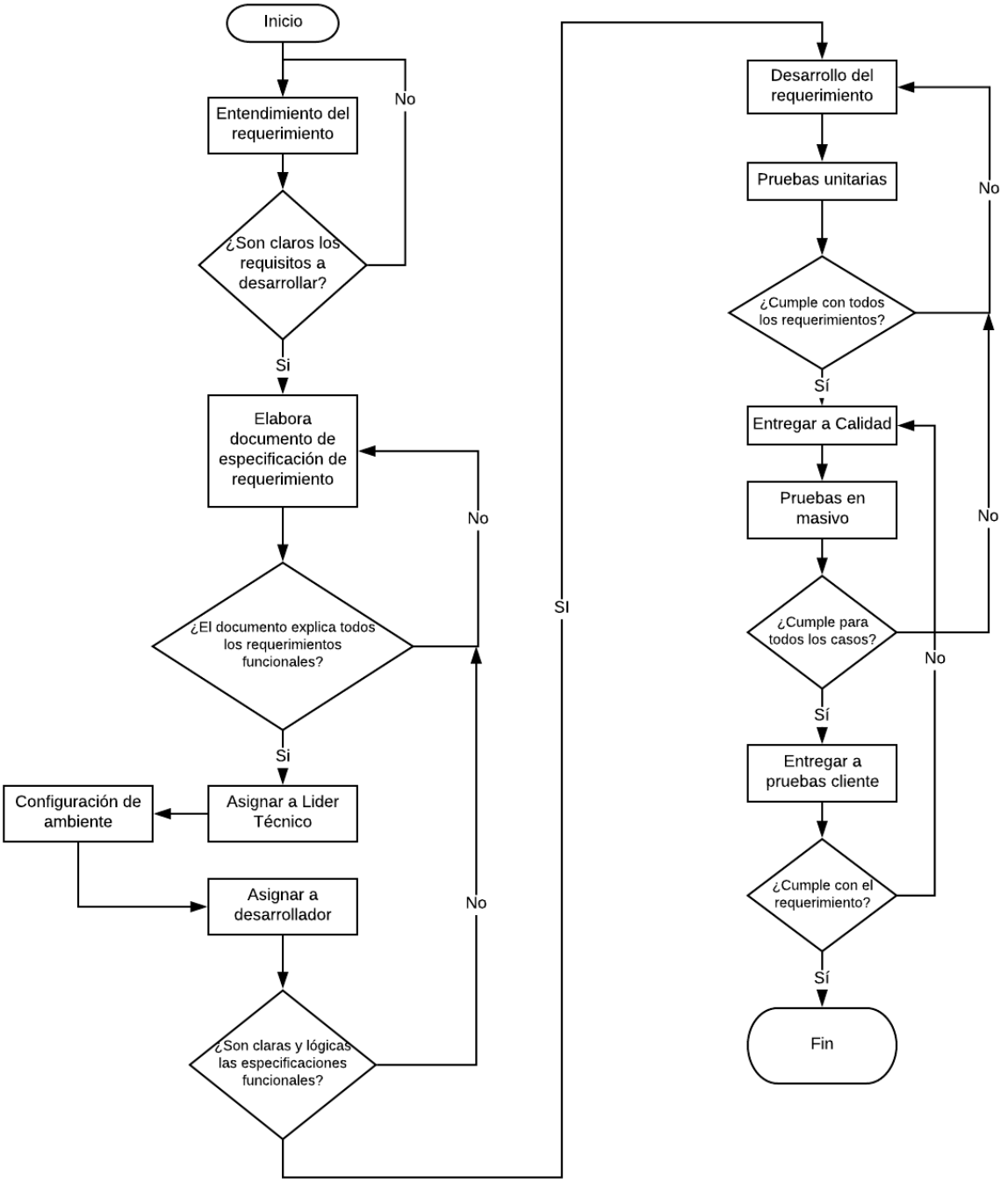

Figura 15: Diagrama de flujo del ciclo de vida de un requerimiento

6. ANÁLISIS DE RESULTADOS

Debido que asumiendo el rol de desarrollador de estrategias de toma de decisión crediticia estamos ubicados en el foco del proceso de elaboración de la solución de software, se logró adquirir experiencia relacionada a todas las fases del desarrollo de software.

Dentro de la etapa de análisis de requisitos, toda vez que se le es asignado al desarrollador un requerimiento este debe realizar una lectura concienzuda al documento de especificación de requerimientos y en dado caso de encontrar fallas, dudas o ambigüedades en el mismo se debe solicitar aclaración de términos y actualización del documento de especificación; En la etapa de diseño y arquitectura durante la elaboración de los formularios de entrada/salida, en la etapa de desarrollo llevando todas las especificaciones a lenguaje máquina en la herramienta SM - PowerCurve®, en la etapa de pruebas tanto en las de funcionamiento que realiza el desarrollador como en la certificación por QA validando y corrigiendo los hallazgos que estos reportan, en documentación y despliegue llevando el desarrollo al ambiente de QA con los adjuntos de pruebas funcionales correctas y en la etapa de mantenimiento ya que el desarrollador debe estar atento a cualquier nueva modificación que pueda surgir para una solución anteriormente desarrollada.

Se presentó un proceso evolutivo con respecto a la calidad del desarrollo, ya que inicialmente la inexperiencia tanto en la herramienta de desarrollo como en el análisis de los requerimientos generaban errores en los desarrollos que posteriormente se recibían como hallazgos reportados por el área de pruebas QA, lo que reducía los índices de calidad de todo el grupo de desarrollo en general. Toda vez que se obtuvieron conocimientos de análisis de asignaciones y herramientas de mejora para los desarrollos, los índices tanto personales como de grupo de desarrollo fueron incrementando mientras que el número de hallazgos reportados por el área de pruebas disminuía.

Se lograron desarrollar 20 estrategias completamente nuevas como también diversos ajustes a estrategias previamente desarrolladas sobre SM que surgieron durante la vigencia de este proyecto, facilitando así la disposición de más recursos humanos en el proyecto actual que se encuentra Experian DataCredito™ sobre la migración de todas las estrategias fabricadas al motor SM - PowerCurve®..

7. TRABAJOS FUTUROS

Aunque actualmente la disposición del ambiente de desarrollo permite llevar a cabo la configuración de los formularios de entrada y salida describiendo cada campo requerido, esta misma configuración puede llegar a presentarse tediosa teniendo en cuenta que en ocasiones algunas estrategias requieren mostrar como respuesta más de 100 variables diferentes que deben ser manualmente configuradas una a una. Podría desarrollarse una aplicación capaz de generar los archivos XML correspondientes a los formularios de una manera directa y añadiendo en lote todas las variables de salida necesarias, de esta manera se realizaría el proceso de configuración de una manera rápida sin importar la cantidad de campos requeridos.

Además se podría desarrollar un manual o un proceso de inducción normalizado para ayudar los nuevos desarrolladores a reconocer las actividades que se deben realizar y los puntos clave que merecen especial atención durante todo el proceso.

7. CONCLUSIONES

Experian Datacredito™ siendo la más grande central de información crediticia debe cumplir plenamente con las exigencias que se le aplican por ley para poder proveer soluciones integrales en administración de información de sus usuarios. La línea de negocio Decisor por medio de soluciones de desarrollo facilita a diferentes entidades financieras la toma de decisión de otorgamiento crediticio siguiendo las reglas de negocio propias de cada entidad apoyando el desarrollo de distintos sectores en la economía local.

La herramienta de desarrollo **SM – PowerCurve®** ofrece por medio del uso de repositorios compartidos la facilidad para elaborar y gestionar diversos desarrollos de manera simultánea con diferentes usuarios. Asociando las aplicaciones generadas en esta herramienta con desarrollos realizados sobre un servidor web, se obtiene como resultado consultas de otorgamiento de crédito realizadas vía web con respuesta de manera inmediata.

El proyecto **Desarrollo para web service de estrategias de toma de decisión crediticia basado en la herramienta de desarrollo SM – PoweCurve® Strategy Managemet** expone las etapas y herramientas adecuadas para realizar soluciones de software personalizadas de una manera correcta y eficiente.

8 REFERENCIAS

[1] Joaquín Fuentes, Realidad virtual aplicada al tratamiento del trastorno de lateralidad y ubicación espacial, España (2003)

[2] Pragma SA, Análisis, diseño, desarrollo, pruebas y despliegue de software, con los estándares de calidad, proceso y tecnologías usadas en Pragma S.A, Colombia (2012)

[3] Universidad Autónoma de Puebla, Fases del proceso de Desarrollo de Software [Online], Disponible en: https://sistemasvd.wordpress.com/2008/07/05/fases-del-proceso-de-desarrollo-del-software [Accesado 3 Junio 2018]

[4] Universidad Politécnica de Valencia, Proceso de desarrollo de software (2003)

[5] Experian Datacredito™, Strategy Design Studio v2.2 Customer Acquisition Training Course (2015)

[6] IBM Software, IBM Rational ClearQuest Version 8.0 (2011)

[7] World Wide Web Consortium, Web Services Description Language – WSDL [2001)

[8] Congreso de la república, Ley estatutaria 1581, Colombia (2012)

[9] SuperIntendencia de Industria y Comercio, Ley No 1266 de Habeas Data, Colombia (2008).

9 ANEXOS

Funciones aplicables de SM – PowerCurve® Strategy Management

MATH FUNCTIONS

Function	Script Suntax	Mathematical Representation/Note
Abs Absolute Value	ABS(<expr>)	y = abs(expr)
Exec* Execute	EXEC("<class.method>")	This provides the means to execute external Java code. The parameter to the EXEC function is a string containing the fully qualified class name (i.e. including package) and method name. At runtime the class must be available on the classpath.
Exp Exponential	EXP(<expr>)	$y = e^{(expr)}$
Ln Natural Logarithm	LN(<expr>)	y = ln(expr)
Log	LOG(<expr>)	y = log(expr)
Log n Log to base n	LOG(<expr>, <base>)	$y = \log_{base}(expr)$
Power**	POWER(<expr>, <power>)	$y = (expr)^{(power)}$
Random***	RANDOM(<max>)	This produces a number from 0 up to the number you have defined. Any integer positive number can be entered.
Remainder	REM(<expr>, <divisor>)	y = REM (expr, divisor) Divides the value in the expr by the divisor and places the remainder (if any) into y
Round down Round up Round nearest Round nearest down	ROUNDD(<expr>, <multiple>) ROUNDU(<expr>, <multiple>) ROUNDN(<expr>, <multiple>) ROUNDND(<expr>,<multiple) Where multiple is the value to be rounded to	Rounds a value Up/Down or to the Nearest multiple of <multiple> When using round nearest, if the value is mid way between rounding up and rounding down, the default is to round up. When using round near down, if the value is mid way between rounding up and rounding down, the system will round down.
To numeric		Value converters

OTHER FUNCTIONS

Function	Script Syntax	Usage
Exec	EXEC("<class.method>")	Enables the user to call any Java code. The .jar file for the external code must be included on the Java Classpath used by the DA.
Exec script Execute script	EXEC_SCRIPT("<script.filename>")	Enables runtime capability of talking to external client-written scripts
Generate UUID	UUID()	User is able to invoke the unique ID generator. The function does not accept any parameters
Get Environment Variable	GETENV("<variable_name>")	Enables the solution designer to obtain information within the business process flow about the bank operator who operates on the Business Process Server (BPS) and use it for either logging/tracing purposes or for executing Boolean conditions. This function always yields a string, which can be either assigned to a Logical Data Source characteristic or used as one side of a comparison.
Insert	INSERT("<lds>", '<data view embedded array>')	Inserts Logical Data Source characteristic into the embedded array Logical Data Source characteristic
Lookup code	LOOKUPCODE (<code table>, <desc table>, <decision reason code>)	Returns the descriptions corresponding to the given reason codes; looks up each of the reason codes from the specified Decision Reason Code Table and writes the description to the specified description table. Code table = Logical Data Source characteristic containing the list of reason codes. Desc table = Logical Data Source characteristic to write the retrieved descriptions into. Decision Reason Code = The referenced Decision Reason Code Table component to use for the look up.
Merge tables	MERGETABLES (<table1>, <table2>, <decision set>, <output table>, <decision category>, <decision text>, <reason code table>)	Merges the contents of two decision setter tables, using the priority order defined in the decision set to produce a consolidated overall decision.
Reset*	RESET("<lds>" or <char>)	The user has the choice of setting the value of an individual characteristic to Null using <char> or setting the values contained in a logical data source <lds> to Null before running the next DA Call, so that it will contain only the new value(s) to be inserted into the DA Result table.

ARRAY FUNCTIONS

Function	Syntax	Usage
Add element	ADDELEMENT(<dynamic array>)	During execution ADDELEMENT increases the size of the dynamic array by 1. Initializes the new element values to NULL.
Clear elements	CLEAR(<dynamic array>)	Reduces the dynamic array size to zero.
Find	FIND(<array>, <value>, <condition (optional)>, <bound (optional)>) value must be valid for the array type	Returns a number representing the location of the first matching value in the array. <condition> a string representing a condition to be applied to the <value>. The following conditions can be used >, >=, =, <>, <= and ~ (RegEx). The value of this parameter must be enclosed in double quotes, e.g. "<=".
Max	MAX(<array>, <bound>)	Returns the largest value in the array
Merge	MERGE(<array1>, <array2>, <order>, <dest>) order is:A (ascending), D (descending), AR (ascending remove duplicates), DR (descending remove duplicates)	Returns the combined and sorted data from two arrays
Min	MIN(<array>, <bound>)	Returns the smallest value in the array
Size	SIZE(<array>)	Returns a numeric integer representing the size of an array, so you can perform nested looping through groups of data
Set size	SETSIZE (<array> , number)	*SETSIZE is a function to set the magnitude of a dynamic array.)
Sort	SORT(<array>, <order>, <dest>) order is A, D, AR or DR as above	Returns a sorted array
Sum	SUM(<array>, <bound>) Bound is a number or expression that represents the high water mark. E.g. if 'bound' = 5, then the function calculates the sum of the first 5 values in the array.	Returns the sum of numeric values in a given numeric array. To sum over the whole array you must enter the length of the array in 'bound'. For example SUM(' Cheque Account.Current Minimum Balance', 5) This will add the first five elements in the Current Minimum Balance array.

STATISTICAL FUNCTIONS

Function	Script Syntax	Usage
Mean	MEAN(<array>, <bound>)	Returns the average value for the array
Median	MEDIAN(<array>, <bound>)	Returns the middle value in the array (when sorted ascending). Where there are an even number of values the average of the middle two will be returned.
Mode	MODE(<array>, <bound>, <modality>, <modalValues>) See example below for detailed explanation	Returns the most common value in the array
Std Standard Deviation	STD(<array>, <bound>)	Returns the standard deviation (σ_{n-1}) of the values held in the array

Function	Script Syntax	Usage
Mean top tail	MEANTT(<array>, <bound>, <top>, <tail>)	Returns the average of the values held in the array ignoring a number of maximum and minimum entries specified
Mean top tail standard	MEANTT2(<array>, <bound>, <toptailstd>)	Returns the average of the values held in the array ignoring entries falling outside the specified number of deviations from the mean
Std top tail	STDTT(<array>, <bound>, <top>, <tail>)	Returns the standard deviation of the values held in the array ignoring a number of maximum and minimum entries specified
Std top tail dev	STDTT2(<array>, <bound>, <toptaildev>)	Returns the standard deviation of the values held in the array ignoring entries falling outside the specified number of deviations from the mean

Function	Script Syntax	Usage
Normal distribution	NORMSDIST(<expr>)	Uses the cumulative normal probability distribution to return the probability of a result that lies <expr> or more standard deviations above the mean.
Norms inv	NORMSINV(<expr>)	Returns the number of standard deviations above the mean that an observation would have to be in order to have the input probability.

DATE & TIME FUNCTIONS

Function	Script Syntax	Note
Date Change	DATE(<givendate>, <period>) period is number of days as a: constant, numeric characteristic or expression givendate is a Logical Data Source date type	Returns a date relative to a given date by the number of days specified by <period>
Day of week	number = DAYOFWEEK(<date>) date is a Logical Data Source date type. number is a Logical Data Source numeric characteristic	This function returns the day of the week for a given date
Days Between	DAYS(<date1>, <date2>) date1 and date2 are Logical Data Source date types or expressions that resolve to date types	Returns a numeric value representing the number of days between two dates
Is valid date*	ISVALIDDATE(<char> or <year>, <month>, <day>) char is a characteristic that equates to a string or an integer value that is to be tested.	Can check whether a single string or integer is a valid date in the CCYYMMDD format or alternatively it can be used to check whether 3 integers or 3 string data types form a valid date
Months Between	MONTHS(<date1>, <date2>) date1 and date2 are Logical Data Source date types or expressions that resolve to date types	Returns the number of whole months between two dates; if the dates of the months are the same it returns 1
System date	SYSTEMDATE(<const>) const is a: constant, numeric characteristic or expression	Returns a date relative to the current date by the number of days specified by <const>
System date time*	SYSTEMDATETIME("yyyyMMdd HH:mm:ss")	Returns the current date and time in the format
System time*	SYSTEMTIME("HH:mm:ss")	Returns the current time in the format
To date		Value converters

TEXT FUNCTIONS

Function	Script Syntax	Note
Is valid date*	ISVALIDDATE(<char> or <year>, <month>, <day>) char is a characteristic that equates to a string or an integer value that is to be tested. year, month, day are characteristics that must all equate to string or all equate to integer values	Can check whether a single string or integer is a valid date in the CCYYMMDD format or alternatively it can be used to check whether 3 integers or 3 string data types form a valid date
Length	LENGTH(<char>)	This function will accept only string data types. It will count all leading and trailing spaces in addition to the value contained in the characteristic.
Concatenate	STRING(<value1>, <value2>)	This function returns a character string that represents the contents of value1 and value2 concatenated.
Substring	SUBSTRING(<char>, <start> , <length>)	This function returns a character string containing a set of characters out of another string
To date		Value converters
To numeric		Value converters
To string		Value converters
Trim	TRIM(<char>)	This function returns a character string with leading and trailing blanks removed.