UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

REPORTE DE TRABAJO PROFESIONAL

AUTOMATIZACIÓN DE PROCESOS PARA EL MERCADO GLOBAL

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A :

DARÍO FERNANDO URDAPILLETA LEYVA

DIRECTOR DE TESIS:
MARIA DE LUZ GASCA SOTO

2015

Dedicado a mi esposa que constantemente

me ha molestado para terminarlo.

Índice

Introducción

En el proyecto de Automatización de procesos para el mercado global se analizó el negocio actual de la empresa cliente (*Ford*) para agregar una selección automatizada de precios a los productos y así reducir considerablemente la cantidad de procesos que requieren intervención manual, ahorrando dinero en horas hombre. Este proyecto añade una nueva regla a la lógica del análisis de conjunto de facturas.

Los retos se encontraron en implementar un cambio que hace modificaciones durante un proceso demasiado complejo de importación y exportación de auto partes. Se encontraron problemas como la identificación de escenarios que la nueva regla *Multiprice* va a corregir, los cuales tienen como consecuencia que el proceso falle (el proceso se detiene con un error que requiere intervención humana) antes de llegar a la misma regla, así como conflictos que la nueva regla genera a partir de los cambios en la base de datos.

Para este mismo proyecto, se generó una nueva interfaz gráfica que permite al usuario, manipular la información obtenida por la nueva regla *Multiprice* en caso de que ésta falle por falta de información o información inconsistente. Un reto que se presentó al diseñar esta interfaz fue, por ejemplo, usar una plataforma particular de la aplicación donde se encuentra el proyecto, la cual no puede ser modificada, ya que puede impactar otros flujos de negocio, lo cual requirió profundo análisis y depuración de código en JavaScript (JS).

Este escrito está organizado de la siguiente manera:

- Presentación del Proyecto: Aquí se describen los detalles del trabajo profesional divididos en las siguientes secciones: Antecedentes, Objetivos, Contexto, Justificación y Requerimientos del proyecto.
- Capítulo 1: Diseño. En este capítulo se describe el diseño lógico de la solución. Se detalla paso a paso lo que la regla *Multiprice* debe hacer.
- Capítulo 2: Interfaz Gráfica. En este capítulo es donde se explica la interfaz gráfica según el diseño lógico de la nueva regla.
- Capítulo 3: Implementación. Se trata acerca de la implementación del proyecto, en este apartado mencionan las dificultades encontradas durante el proceso de implementación y la soluciones propuestas y aplicadas a cada una de ellas.
- Capítulo 4: Resultados obtenidos. Se muestran los resultados obtenidos durante el proyecto y los beneficios recibidos tanto por parte de las áreas de operaciones como la misma área de desarrollo.
- Finalmente se presentarán las Conclusiones y Bibliografía de este trabajo.

Durante el periodo laboral iniciado el 25 de noviembre del 2013 y concluido el 3 de marzo del 2015 se realizaron diversas actividades como lo son:

- Documentar proyectos en *IBM – DataStage*.
- Distintos cambios menores y mantenimiento a aplicaciones como *Tradesphere Importer* (TSI – maneja los procesos de importación de productos), *Tradesphere Exporter* (TSE –

maneja los procesos de exportación de los productos), *Tradesphere Classifier* (TSC – clasifica los productos en las bases de datos de TSI y TSE según sea el caso).

- Se generó un ETL (transferencia de datos) desde una base de datos Oracle a una *SQL Server* usando la aplicación *SSIS* de *Microsoft*.
- Se tomó la gestión del proyecto web *Track & Trace* ya iniciado con el desarrollo por parte de un proveedor (SPAN) de India. El proyecto monitorea el proceso aduanal de punta a punta.

Presentación del Proyecto

Se describirá brevemente la historia de la compañía y los proyectos relevantes que dieron lugar al proyecto tratado en este documento.

Se presentará el objetivo general del proyecto, explicaremos la importancia del proyecto para la empresa y por último los requerimientos que fueron recibidos por parte del área de Operaciones.

I. Antecedentes

Livingston International es una empresa de origen canadiense, cuya historia se remonta desde 1945 cuando la firma de Gerry Livingston se incorporó al registro como *Livingston Lumber and Manufacturing Ltd*. En 1986, la división de gestión aduanal fue creada con el nombre de *Livingston International*.

En 1993 expandió su mercado a los Estados Unidos convirtiéndose en la primera gestora aduanal que proporcionaba servicios de costa a costa en ambos países, Canadá y Estados Unidos de América.

Posteriormente, *Livingston International* le compró al banco *JPMorgan* el negocio de "Conciliación de Mercado Global" antes conocido como *Vastera* extendiendo sus operaciones a México, Europa y Asia, [3].

Vastera Inc. es un proveedor de servicios Web y productos, los cuales simplificaban y automatizaban bienes de importación y exportación en una escala global. Se originó en el estado de Virginia en 1991 con el nombre de *Export*

Software International Inc. y posteriormente cambiando su nombre a *Vastera Inc.* en 1997, [3].

Cuando *Livingston International* compró a *Vastera*, ésta pasó a ser el área de Mercado Global (*Global Trade Market* – GTM).

Su negocio es la información referente al mercado global, el cual tiene una extensa concentración de distintas reglas aduanales para cada país, ésta se usa para automatizar los procesos del mercado global. Los clientes le compraban a *Vastera* una licencia usando su servicio por medio de un portal web o bien le otorgaban a *Vastera* el control de sus operaciones directamente.

Vastera tiene una gran variedad de productos, entre ellos:

- TSI (*TradeSphere Importer*). Maneja todos los procesos de importación, por ejemplo, cuando una empresa necesita importar mercancía en el proceso, utiliza esta aplicación para iniciarlo desde enviar la factura recibida por el proveedor, hasta enviar ésta al agente aduanal.
- TSE (*TradeSphere Exporter*). Maneja todos los productos que se enviarán al extranjero de la misma forma que TSI, es decir, utilizando la aplicación para dar seguimiento al producto.
- TSC (*TradeSphere Classifier*). Maneja la clasificación de mercancía y propaga la información de cada producto a las distintas aplicaciones de *Vastera* como TSI y TSE.

TSI recibe la información de las facturas del proveedor. Éstas se almacenan en una base de datos, pasando por un proceso de validación de datos que revisa que los datos

tengan lo mínimo necesario y sean consistentes para ser enviados al agente aduanal y que pueda pasar satisfactoriamente por el proceso aduanal.

Transaction Automation (TA) es un proyecto cuyo objetivo principal es automatizar el proceso de importación, de tal forma que la necesidad de intervención humana se reduzca lo más posible. Constantemente se generan requerimientos para adecuar el proyecto basados en estadísticas de los errores o situaciones que ocasionan mayor necesidad de intervención.

Ford es el cliente más importante que maneja el área de Mercado Global y es el cliente que financió el proyecto de TA.

El proceso de análisis es llamado *Consolidated Rated Invoice Analysis* (CRIA) se encarga de revisar toda la información recibida y completarla con información ya existente de los productos en cuestión. Ésta a su vez, alerta a los usuarios tanto en las fallas como en su correcto desarrollo, con el motivo de enviar la información correctamente lo antes posible.

Unos de los valores que el CRIA agrega a la información de las facturas son: el precio del producto y el tipo de moneda asociado al precio. Aunque el tipo de moneda siempre es el mismo dado el origen de la factura (país de procedencia), el precio varía según el proveedor y el destino. De esta manera proveedores distintos tienen precios distintos, así como los costos de envío varían dependiendo de la distancia.

Antes de este proyecto, se tenía una variable booleana para indicar que se pueden tener distintos precios para el producto, detener el proceso e indicar la necesidad de intervención manual por parte del usuario operativo. Esto ocasionaba costo de operación que obliga a arreglar los datos

manualmente. Por esto surge la necesidad para reducir el número de facturas que necesitan atención manual y consecuentemente se creó el proyecto de *Multiprice*.

II. Objetivo General

El objetivo general de este proyecto de reporte de trabajo profesional es el generar una solución dinámica que ocupe la menor intervención humana posible, para localizar el precio correcto de los productos de un conjunto de facturas (*Consolidated Rated Invoice* - CRI) durante el proceso de análisis de conjunto de facturas (*Consolidated Rated Invoice Analisis* - CRIA).

Para lograr lo anterior se requiere una regla que permita obtener el precio correcto del producto, o bien, alertar al usuario operador si no es posible asegurar que el precio es correcto para que se resuelva el problema mediante la intervención manual.

Cabe mencionar que dicha intervención será imposible de eliminar completamente, ya que requiere que la información proveniente del cliente sea correcta siempre, por lo que es suficiente minimizar la interacción lo más posible. A su vez, se requiere que la intervención humana sea lo más sencilla y le cueste al usuario operativo el menor tiempo posible.

Como consecuencia de la implementación de esta regla, se creó una interfaz gráfica que permite al usuario operativo de manera simple manipular los precios para que en futuras ocasiones la intervención manual no sea necesaria para un caso similar.

III. Justificación del Proyecto

Dentro de la información del cliente FORD se tienen dos clasificaciones de productos dado el precio. El primero es aquel producto cuyo precio no varía dependiendo del proveedor (precio fijo). El segundo es aquel que, dependiendo del proveedor, el precio puede variar (precio variante).

Originalmente se identificaba la clasificación a la que el producto pertenecía; si era un precio variante, entonces se alertaba al usuario operativo como si fuera un error (se le entiende como "fallo" (*fail*) a detener el proceso y enviar una alerta con mensaje de error) y requerir la intervención manual. Los procesos fallados son regresados al estado original del registro antes de realizar el proceso.

La directora del área de Operaciones, dentro de sus datos estadísticos, notó que el porcentaje de productos con precio variante era alto, por lo tanto, la intervención manual representaba un costo importante en la empresa. Por esta razón se decidió corregir el módulo para encontrar el precio correcto en caso de que éste exista, y sólo detener el proceso automático en el caso de que el producto tenga precio variante y no se pueda asegurar encontrar el precio correcto.

Para los casos que de igual forma requieren intervención manual, se planeó una estrategia para no repetir este proceso en situaciones similares, es decir, situaciones en las que el proveedor, origen y destino sean los mismos y no se haya registrado un cambio de precio en el producto.

Resulta fundamental para este proyecto el desarrollo de este módulo ya que permitirá reducir las horas-hombre del área de Operaciones. Considerando que los datos provenientes del

cliente no son siempre completos, se requiere construir una lógica que considere distintas configuraciones de datos no esperadas.

IV. Contexto

TradeSphere es una plataforma Web creada por *Vastera* como plataforma única de sus productos de tal forma que se puedan adecuar a las distintas necesidades de negocio. Hoy en día necesita una reestructuración en la parte gráfica dados los cambios que han sufrido las tecnologías Web y funciona exitosamente desde 1999. La plataforma está basada en HTML[1] con *JavaScript* [2]para la parte gráfica y *Java*[3] tanto para la parte lógica como para la comunicación con la base de datos. Esta arquitectura de tres capas se encontraba inicialmente en un servidor basado en la plataforma *Windows*, sin embargo a finales del 2013 se migró a un servidor *Linux* debido a que los últimos son más seguros, estables y las tecnologías de la plataforma no están dentro de las limitantes de *Linux* (como ASP.NET[4]). *TradeSphere* utiliza una plataforma propia de programación llamada *Colaborative Platform* (CP) la cual maneja todas las interacciones lógicas con la base de datos, archivos, seguridad, entre otras y promueve al equipo de programación a usar sus clases y funciones que ya están probados y siguen buenas prácticas de programación. Esto permite a los programadores a enfocarse en su tarea y no

[1] HTML, sigla en inglés de *HyperText Markup Language* (lenguaje de marcas de hipertexto) hace referencia al lenguaje que incorpora etiquetas o marcas que contienen información adicional acerca de la estructura de su texto o su presentación usado para la elaboración de páginas Web.
[2] *JavaScript* es un lenguaje de programación interpretado que se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente implementado por parte de un navegador Web.
[3] Java es un lenguaje de programación de propósito general, concurrente y orientado a objetos que fue diseñado para permitir a los programadores a escribir el programa una vez y que lo ejecuten en cualquier dispositivo.
[4] ASP.NET es un *framework* para aplicaciones Web desarrollado y comercializado por *Microsoft*.

tener que resolver problemas de comunicación con la base de datos o implementación de las estructuras de datos existentes.

El Importador *TradeSphere* (*TradeSphere Importer* - TSI) es una aplicación dentro de la plataforma *Tradesphere* que se encarga de manejar todos los procesos de importación. Uno de estos procesos es el análisis de conjuntos de facturas (*Consolidated Rated Invoice Analysis* – CRIA).

El CRIA es un proceso basado en una serie de reglas bajo cierto orden administradas por *TradeSphere*. Se encarga de analizar el contenido capturado de los conjuntos de facturas (*Consolidated Rated Invoices* - CRI) y asegurar que sean correctos para el proceso de importación. Revisa distintos campos en el CRI como el precio, el origen, cantidad y los productos incluidos en cada una de las facturas. Al final del proceso se genera un reporte mostrando los errores o advertencias generados por las reglas. El proceso se puede activar manualmente con uno o más conjuntos de facturas o bien, automáticamente cada vez que TSC agrega una nueva factura a la base de datos.

Las reglas del CRIA relevantes para este documento son:

- *FORD Product Subpull Rule*: Obtiene toda la información disponible para el CRI.
- *FORD Post-Product Subpull Rule*: Valida que la información del CRI esté correcta.
- *Calculation Value Rule*: Calcula el precio total de una factura dado el precio unitario y la cantidad.
- *Multiprice Rule*: Falla el proceso si un producto puede tener distintos precios, lo cual requiere de atención manual.

La regla *Multiprice* [5]requiere la atención manual para los productos que tienen distintos precios. Éste es un caso común y resulta ser un proceso costoso en términos de operatividad. Por esta razón se generó un requerimiento para eliminar esta regla y sustituirla por una que maneje de otra forma los distintos precios.

La base de datos tiene un diseño jerárquico, en la cual las tablas que representan una especialización de otras tienen como llave primaria a las llaves del padre más su propio conjunto de llaves. Tiene como desventaja la repetición de información, pero su gran ventaja es que hace más eficiente la búsqueda de datos reduciendo la carga de procesamiento a la base de datos y reduciendo los tiempos de respuesta.

Los nombres de las tablas de la base de datos contienen la jerarquía del diseño. Por ejemplo, en *TradeSphere Importer* la tabla del conjunto de facturas (CRI) se llama: TSI_CRI y la tabla de las facturas (*invoice* - INV) que pertenecen a un conjunto de facturas es TSI_CRI_INV.

Las entidades involucradas en la solución son las siguientes:

- Producto (TSI_PD): Tiene la información de cada producto que el cliente (FORD) maneja.
- Versión del Producto (TSI_PD_V): Tiene la información de una versión en particular de un producto la cual depende del vendedor.
- Datos del producto (TSI_PD_V_DATA): Tiene la información particular de la versión del producto. El valor de la versión cambia con el tiempo y un registro de esta tabla define el estatus del producto en un momento específico.

[5] La regla *Multiprice*, depende de la regla *Ford Product Subpull Rule*.

- Precio del producto (TSI_PD_V_PRICE): Tiene la información más actualizada del precio del producto. Esta tabla no existía originalmente y se creó durante la solución presentada.
- Conjunto de facturas (TSI_CRI): Tiene la información de un proceso aduanal que puede tener una o más facturas con productos que se trasladan de un país a otro.
- Factura (TSI_CRI_INV): Tiene la información de una factura en específico.
- Sitio del proveedor (TSI_CRI_INV_PSITE): Tiene la información del origen de los productos de la factura así como la información del destino.
- Elemento de la factura (TSI_CRI_INV_ITM): Tiene la información de un producto de la factura como la cantidad de ejemplares del producto.

V. Requerimiento

El diseño original de las reglas del CRIA no maneja de manera correcta los precios variables de los productos de FORD que dependiendo del proveedor y del destino del producto pueden cambiar. Este diseño obtiene el último precio ingresado el cual puede ser o no el precio indicado. Si el producto tiene precio variable, se produce un error en el proceso para indicar que se requiere una intervención manual.

Durante la fase de soporte de producto se ha agregado la información de las distintas variaciones de precios por cada fuente/destino:

1) TSI debe asegurar que los precios de los productos con variaciones que dependen de la fuente/destino sean correctos o bien, puestos en espera para tener una

intervención manual, con el fin de minimizar el riesgo de enviar precios incorrectos.

2) TSI debe buscar el precio considerando los valores fuente/destino del envío y así guardarlos en una nueva tabla de la base de datos que debe ser actualizada constantemente.
3) TSI debe identificar la diferencia entre los precios que se encuentran de forma automática y los que se determinan de forma manual, para dar prioridad a estos últimos.
4) TSI debe permitir a los usuarios poner el valor del precio de forma manual.

Con esto se busca reducir la necesidad de intervención manual y agilizar el proceso y, por tanto, se reducen los costos y se incrementa la satisfacción del cliente (FORD).

Capítulo 1. Diseño de la solución

Durante todo el proceso, el diseño de la solución cambió constantemente debido a la falta de un buen análisis y generación de requerimientos. Es importante tener bien definidos los requerimientos del problema ya que ocasionan que se tenga que regresar a la fase de diseño de la solución al descubrir que la solución obtenida no es la esperada. Este problema se debió a la reciente compra del área de *Vastera* y el despido de la mayor parte del equipo de análisis y desarrollo, lo cual produjo una carencia en expertos en la plataforma.

El diseño abarca dos partes:

1. La regla debe seguir la lógica requerida para reducir la necesidad de intervención manual.
2. Construir una interfaz de usuario sencilla de usar, dentro de la plataforma *TradeSphere*, para que el analista pueda corregir el precio manualmente al identificar el error producido.

1.1 Copia de la regla *Multiprice* (*MultipriceCopyRule*)

La regla *Multiprice* se encargaba de detener el proceso con un error en los casos que el producto podía tener precios múltiples. La regla determinaba si era el caso por medio de una variable lógica dentro de la tabla del producto (TSI_PD). Conforme el departamento de operaciones comenzó a enviar cada vez más productos con precios múltiples, esta regla se convertía en un proceso costoso y lento.

La solución elimina completamente la lógica de la regla *Multiprice* y la reemplaza con una nueva regla: *MultipriceCopyRule*.

Datos de Entrada/Salida de la regla *MultipriceCopyRule*

A continuación se describirán las distintas variables de entrada y salida que componen la lógica de la nueva regla *MultipriceCopyRule*.

Valores de entrada por cada CRI.

- SUB_ORG: Se refiere al cliente, en este caso siempre será FORD.
- CTRY_CODE: Es el país de origen del CRI.
- CRI_NAME: Se refiere al nombre identificador del CRI.
- VERSION_NUM: Este valor indica la versión del CRI.

Valores de salida calculados por cada producto en cada una de las facturas en el CRI.

- SUB_ORG: Se refiere al cliente, en este caso siempre será FORD.
- TSI_PROD_ID: Hace referencia al código de producto.
- V_ID: Es el identificador del proveedor en la tabla de producto usado internamente para localizar al registro del precio.
- SHIP_FROM: Se refiere al código del proveedor en la factura.
- SHIP_TO: Es el código del destino.
- DATA_VALUE: Se refiere al precio del producto.
- ISO_CURR_CODE: Indica el tipo de moneda.
- CREATED_DATE: Es la fecha en que fue creado el registro.
- CREATED_BY: Se refiere al usuario o programa que creó el registro.

- MODIFIED_DATE: Es la fecha en que fue modificado el registro por última vez.
- MODIFIED_BY: Es el usuario o programa que modificó el registro por última vez.

1.2 Proceso de la regla MultipriceCopyRule

En esta sección se detallará el proceso que compone la regla *MultipriceCopyRule*. Al iniciarse la regla *MultipriceCopyRule*, se consigue la información del CRI por medio de un tipo de datos basado en la plataforma colaborativa (CP). Este objeto contiene la información de la tabla TSI_CRI y todos sus hijos (Para mayor referencia, ver la Figura 17 del Anexo).

Ciclo de facturas

Dentro de un CRI se encuentran una o varias facturas, las cuales se analizan individualmente debido a que tienen proveedores y destinos distintos. Por cada Factura (TSI_CRI_INV) se consigue la siguiente información:

- TSI_CRI_INV.INV_NAME: Se refiere al nombre de la factura del conjunto de facturas analizado.

Por cada factura se buscan en la tabla TSI_CRI_INV_PSITE los registros que cumplan con el siguiente criterio:

- Se define como el proveedor (SHIP_FROM) al campo TSI_CRI_INV_PSITE.SITE_ID cuando TSI_CRI_INV_PSITE.PTNR_TYPE = 'SUPPLIER'.
- Se define como el destino (SHIP_TO) al campo TSI_CRI_INV_PSITE.SITE_ID cuando TSI_CRI_INV_PSITE.PTNR_TYPE = 'ULTIMATE_CONSIGNEE'.

- Se define como **description** a la concatenación de **SHIP_FROM** y **SHIP_TO**. Es decir, **SHIP_FROM+SHIP_TO**

Ciclo de productos

Cada factura tiene uno o varios productos que se importan. Se obtiene la lista de productos que se encuentran en cada factura y por cada uno de estos productos se realiza lo siguiente:

- Se obtiene el identificador del producto en la factura con la variable **TSI_CRI_INV_ITM.TSI_PROD_ID** el cual debe coincidir con la variable **TSI_PD.TSI_PROD_ID** en la tabla de productos **TSI_PD**.
- Se obtiene el identificador del proveedor del producto con la variable **TSI_CRI_INV_ITM.VEND_PTNR_ID** la cual debe coincidir con el vendedor de la tabla **TSI_PD_V**.
- Utilizando el método **checkMultiPriceForProduct()** (Figura 18 del Anexo) obtenemos la variable **multiprice_flag**, la cual indica si el producto tiene o no varios precios. Este método obtiene la variable **TSI_PD.SYS_VARCHAR_1** y regresa verdadero si el valor es "Y" y falso en cualquier otro caso.

Si las variables **SHIP_FROM** y **SHIP_TO** obtenidas en el ciclo de facturas son nulas, se verifica la variable de **multiprice_flag** para revisar si el producto tiene distintos precios. Si se trata de un producto que acepta varios precios, se detiene el proceso de la regla con el error 9728 que indica que un producto que acepta precios múltiples debe tener los valores de proveedor y destino. Si no permite distintos precios, entonces no será necesario tener los valores de **SHIP_FROM** y **SHIP_TO** siguiendo la lógica anterior que es encontrar el

precio distinto a 0 más reciente y, en el caso descrito, las variables SHIP_FROM y SHIP_TO se cambian al valor estático 'ALL'.

Ya sea teniendo los valores de SHIP_FROM y SHIP_TO originales o reemplazados por 'ALL', se procede a buscar la información del producto.

Obtener la información del producto

Una vez encontrada la información que conforma el valor de description, se obtiene la información del producto. La información que se obtiene de la tabla de historial (TSI_PD_V_DATA) es la siguiente:

- TSI_PD_V_DATA.DATA_VALUE: Se refiere al precio de la tabla PD_V_DATA la cual contiene el historial de precios de la versión del producto.
- TSI_PD_V_DATA.USER_VARCHAR_1: Se refiere al tipo de moneda de cambio dentro de la tabla de historial.
- TSI_PD_V_DATA.USER_DATE_3: Se refiere a la fecha en que se modificó por última vez el registro del precio del producto.
- TSI_PD_V_DATA.V_ID: Se refiere al identificador del vendedor para localizar el mismo precio de nuevo y encontrar o generar el registro en la tabla nueva de precios.

Si las variables SHIP_FROM y SHIP_TO no son nulas, se busca en el historial de precios al precio más reciente que concuerde con description (SHIP_FROM+SHIP_TO). Se debe asegurar también que los resultados nulos se cambien por 0 ya que de lo contrario el orden del resultado de la petición a la base de

datos no será el deseado y que la fecha sea la más reciente de las fechas que coincidan. La petición a la base de datos debe tener un orden descendente en el campo del precio para asegurar elegir un valor que no sea 0. Debido a un error que no se ha resuelto en el proceso de actualización de precios, existen registros del historial de precios con la misma fecha y con precio igual a cero y por esta razón es necesario el ordenamiento con respecto al precio.

Si no se encuentra un valor mayor a cero del precio del producto, se verifica si permite varios precios. De ser así, se realiza el proceso de validación de vendedor; en caso contrario, se busca el precio sin considerar el valor de description. El Área de Operaciones pidió que se intentara conseguir el precio con el valor de description antes de verificar el valor de multiprice_flag en caso de que algún producto tenga mal dicha bandera. La prioridad era el precio donde el valor de description de la factura empatara con el del historial de precios.

Independientemente de si se encontró esta información o no, es necesario conseguir el identificador de un vendedor (V_ID) válido, ya que la tabla encargada de concentrar los precios actuales (PD_V_PRICE), tanto provenientes del historial o modificados manualmente, necesita un V_ID para tener un registro en la tabla.

Así que si no se encontraron precios en el historial, se usa el método getVIDFromProduct() el cual busca el valor de V_ID correspondiente al producto dado el TSI_CRI_INV_ITM.VEND_PTNR_ID. La Figura 19 ilustra el diseño lógico del método. En caso de no encontrarlo se detiene el proceso de la regla y se muestra el error 9729 que indica que no hay un vendedor válido para el producto.

Obtener la información del precio en la nueva tabla

La nueva tabla generada para esta regla es TSI_PD_V_PRICE, ésta se define como la información más reciente de precios asociados con un vendedor, o bien, no asociados en caso de no haber uno específico. Es indispensable mantener esta definición ya que el precio puede variar por los siguientes procesos:

1. Un nuevo proceso de validación del CRI siempre y cuando el precio se ha actualizado en el historial de precios (TSI_PD_V_DATA) y es un precio mayor a 0.
2. Una modificación manual del Analista de Mercado.

La regla *MultipriceCopyRule* se encarga de hacer la primera modificación durante el proceso de validación del CRI. Pero la segunda puede ocurrir en cualquier momento por medio de la interfaz gráfica de TSI ya que es una intervención manual.

Para asegurar que la regla *MultipriceCopyRule* mantenga la definición de la nueva tabla con la información más reciente de precios (TSI_PD_V_PRICE), se tiene que conseguir la información de esta tabla para compararla con la obtenida en el historial de precios (TSI_PD_V_DATA).

Se obtiene la siguiente información asegurando que los valores de las variables TSI_PROD_ID, V_ID, SHIP_FROM y SHIP_TO sean los mismos:

- TSI_PD_V_PRICE.DATA_VALUE: Se refiere al precio del producto dependiente de los valores de SHIP_FROM y SHIP_TO
- TSI_PD_V_PRICE.USER_DATE_2: Se refiere a la última fecha en la que fue modificado el registro.

- TSI_PD_V_PRICE.ISO_CURR_CODE: Se refiere al tipo de cambio de moneda del precio del producto.
- TSI_PD_V_PRICE.USER_VARCHAR_1: Se refiere al usuario que modificó por última vez el registro.

En el caso de no encontrarse el registro, se guarda en la tabla con el precio más reciente (TSI_PD_V_PRICE) un nuevo registro con los valores de precio de la tabla de historial.

Comparación de valores

Una vez obtenidos los valores de ambas versiones de los precios se efectúa una decisión sencilla para definir cuál precio debe mantenerse en la tabla con el precio más reciente (TSI_PD_V_PRICE). Los criterios de decisión son los siguientes:

- Si ambos precios son distintos de 0 y la tabla del historial es más reciente, entonces el valor que se conserva es el valor de la tabla del historial (TSI_PD_V_DATA). El proceso de la regla termina sin registrar errores.
- Si ambos precios son distintos de 0 y el historial es el menos reciente, entonces el valor que se conserva es el valor de la tabla del precio más reciente (TSI_PD_V_PRICE). El proceso de la regla termina sin registrar errores.
- Si el valor del historial es mayor a 0 y el otro es 0 entonces el valor que se conserva es el valor de la tabla de historial (TSI_PD_V_DATA). El proceso de la regla termina sin registrar errores.

- Si el valor del historial es 0 y el otro valor es mayor a 0, entonces el valor que se conserva es el valor de la tabla del precio más reciente (TSI_PD_V_PRICE). El proceso de la regla termina sin registrar errores.
- Si ambos valores son 0, entonces se detiene el proceso de la regla y se registra el error 9727 que indica que no hay un precio mayor a cero para el producto con el vendedor (V_ID) determinado.

Durante este proceso, se actualizan los siguientes campos de la base de datos con los valores finales para que el resto de las reglas continúen su trabajo con los valores de los precios más recientes.

- TSI_CRI_INV_ITM.PD_HTS_UNIT_PRICE que se refiere al valor final del precio del producto en la factura.
- TSI_CRI_INV_ITM.PD_HTS_IS_CURR_CODE que se refiere al valor del tipo de cambio de moneda del precio del producto en la factura.

Con esto termina la descripción del proceso de la regla *MultipriceCopyRule* la cual asegura que el precio de los productos de las facturas del conjunto de facturas analizados sean los más recientes y mayores a cero en caso de existir en el historial de precios o haber sido modificados manualmente.

Capítulo 2. Interfaz gráfica

La interfaz gráfica de usuario es un programa que está compuesto por objetos gráficos como íconos, botones, dibujos, modelos 3D e imágenes y por acciones disponibles que el usuario puede tomar con los dispositivos de entrada/salida del equipo que está utilizando.

Como fue mencionado anteriormente, uno de los procesos para actualizar el precio de los productos es una intervención manual por medio del analista en los casos que el precio no se pueda determinar automáticamente. Para esto, se debe desarrollar una interfaz de usuario que permita manipular la información de la tabla TSI_PD_V_PRICE, la cual debe ser sencilla y rápida de usar.

A continuación se describirá la interfaz gráfica desarrollada para el sistema.

Pantalla principal

Esta pantalla será la primera pantalla que el usuario visualice y permitirá realizar cualquiera de las funcionalidades planeadas para la interfaz gráfica. Sirve como punto de partida para cada tarea que quiera tomar el usuario.

El usuario entra a la sección para modificar la tabla de precios ("*Price by Supplier*") por medio de un menú en la plataforma *Tradesphere Importer*. El usuario de esta forma accederá a la pantalla principal del módulo, la cual se muestra a continuación en la Figura 1:

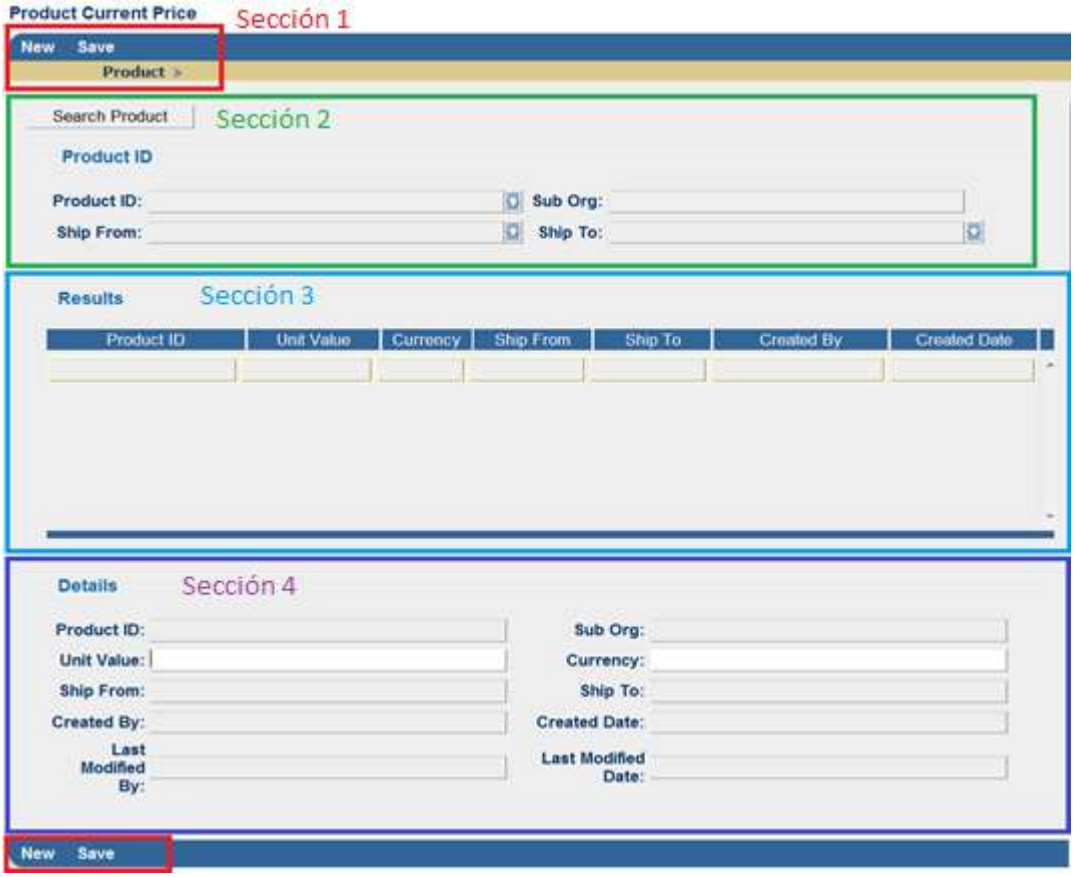

Figura 1. Pantalla principal del módulo "*Price by Supplier*"

La pantalla principal se puede descomponer en cuatro secciones para facilitar su entendimiento.

Sección 1 - Gestión de elementos de Tradesphere

Ésta es una sección predeterminada por el sistema de *Tradesphere*. Todas las pantallas que se encargan de manipular registros de la base de datos en *Tradesphere* tienen las funcionalidades de almacenamiento y de creación de un registro nuevo. A continuación se explican cada uno de los elementos de esta sección:

- El botón *New* (nuevo), que se encuentra del lado izquierdo, en los marcos superior e inferior de la pantalla principal, se encarga de reestablecer los datos de la pantalla principal obteniendo el estado inicial de la misma. Si existen rastros de un registro nuevo o uno modificado, la interfaz mostrará un mensaje al usuario final preguntándole si se quieren guardar los cambios antes de continuar:

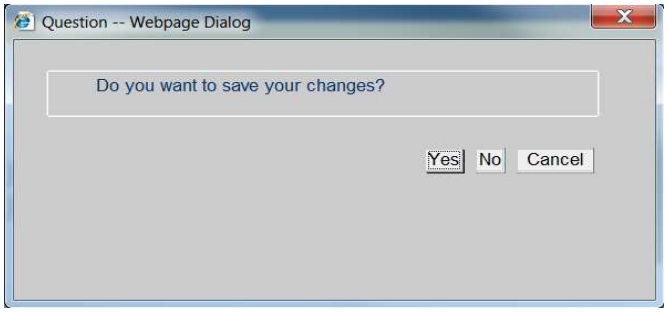

Figura 2. Mensaje de confirmación para guardar los cambios.

Las opciones que se listan a continuación le permitirán al usuario principal continuar con el flujo de la tarea que está realizando y se muestran en la Figura 2.

  - Si se elige la opción de [*Yes*] (sí), ésta almacenará los cambios realizados y regresará a su estado inicial.
  - Si se elige la opción de [*No*] ésta regresará a su estado inicial sin guardar los cambios.
  - Si se elige la opción de [*Cancel*] (Cancelar) ésta cancelará el proceso de la tarea del botón de *New*.

- Como se puede observar en la Figura 3, al oprimir el botón *Save* (almacenar) se guardarán los datos

considerando su estado actual y se pueden observar en la parte de abajo del navegador los siguientes mensajes:

Figura 3. Mensajes en el explorador que indican el estado del proceso de guardado.

La Figura 3(a) se muestra cuando el estado del proceso de almacenamiento se encuentra en proceso. La Figura 3(b) indica que el proceso ha terminado.

Inmediatamente debajo de las opciones de *New* y *Save* en la sección superior de la interfaz se observa una jerarquía de contenidos. En el caso de la Figura 1, se visualiza el texto *Product* (Producto). Es una funcionalidad predeterminada de *Tradesphere* que se permite debido a la estructura de la base de datos. Estos textos se usan para navegar entre las distintas tablas del registro. Para este caso, sólo se maneja una tabla por lo que sólo hay un texto.

Sección 2 – Búsqueda de producto

La sección 2 de la Figura 1 permite la búsqueda de productos para consultar o modificar en el resto de la interfaz gráfica. La Figura 4 muestra un botón de búsqueda, un campo estático y tres criterios de filtro que a continuación se describen:

Figura 4. Sección de la interfaz gráfica que permite iniciar el proceso de búsqueda de productos.

- Al presionar el botón [*Search Product*] (buscar producto) se muestra una ventana emergente que le permite al usuario final buscar un registro único que contiene los tres criterios de filtro posibles.
- Al presionar el botón que se encuentra junto al campo de *Product ID* (El identificador del producto) se muestra la ventana emergente que permite al usuario buscar todos los productos que tengan el mismo código identificador de producto.
- Al presionar el botón que se encuentra junto al campo de *Ship From* (enviado desde), que representa al proveedor, se muestra una ventana emergente que permite al usuario buscar todos los productos que tengan asociado al mismo proveedor.
- Al presionar el botón que se encuentra junto al campo de *Ship To* (enviado a), que representa al destinatario se muestra la ventana emergente que permite al usuario buscar todos los productos que tengan asociado al mismo destinatario.
- El texto Sub Org (sub-organización) muestra a la empresa a la cual pertenece el producto que, por el momento, siempre será FORD hasta que se incluya este desarrollo para otras compañías.

Cualquiera de las opciones anteriores de búsqueda puede regresar diversos productos como resultado. En caso de que se requiera sólo un producto, el botón de búsqueda (*Search Product*) permite utilizar los tres campos *(Product ID, Ship From* y *Ship To*) debido a que la llave de un producto en la base de datos se compone de estos tres campos.

Sección 3 – Resultado de búsqueda

Esta sección muestra uno o más resultados de la búsqueda. El usuario podrá dar clic a uno de ellos y cambiará el contenido de la sección 4 que muestra los detalles del producto y se describe adelante.

Una vez que se haya elegido el registro o los registros y se selecciona el botón *OK* en la ventana emergente, ésta desaparece y los registros seleccionados se muestran en la tabla de resultados como se muestra en la Figura 5.

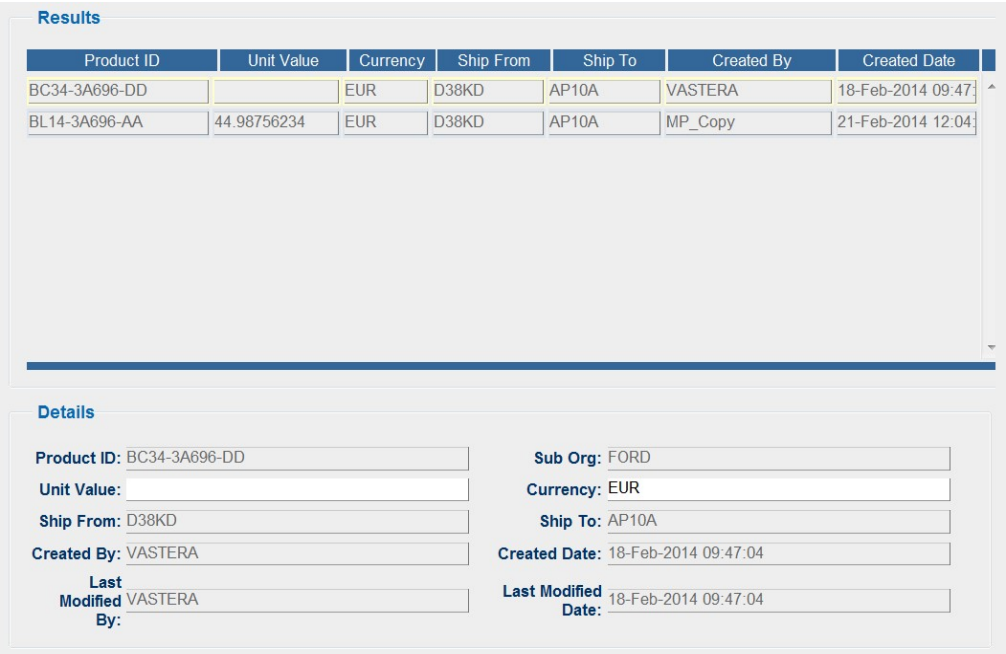

Figura 5. Pantalla para elegir el producto a modificar.

En la Figura 5 se observan los campos relevantes que describen e identifican al producto. A continuación, se describen estos campos:

- *Product ID* (Identificador del producto): Es la cadena que identifica al producto en todo el proceso de exportación e importación.
- *Unit Value* (Valor unitario): Se refiere al costo del producto.
- *Currency* (Tipo de moneda): Indica el tipo de moneda en que el valor unitario del producto está representado.
- *Ship From* (Enviado desde): Se refiere al código de proveedor en la factura.
- *Ship To* (Enviado a): Es el código del destinatario en la factura.

- *Created By* (Creado por): Se refiere al usuario o sistema que registró el producto en la base de datos.
- *Created Date* (Fecha de creación): Es la fecha en que el producto fue registrado en la base de datos.

Por omisión, aparece seleccionado el último registro, esto se observa en la pantalla con un borde de color azul. Para seleccionar otro producto, se deberá dar clic al área que lo conforma, esto actualizará los valores de la Sección 4 para concordar con el producto seleccionado. En la Figura 6 se muestra el cambio de los datos al elegir un producto al seleccionar el producto BC34-3A696-DD y en la sección que muestra los detalles del producto.

Figura 6. Pantalla que muestra el cambio de los datos una vez que se eligió un producto distinto.

Sección 4 – Detalles del producto

En esta sección se muestran los detalles del producto seleccionado en la sección de resultados de la búsqueda. Aquí, el usuario podrá modificar el precio y el tipo de moneda cuando se requiere una intervención manual para actualizar el precio correcto cuando el proceso automático no lo encuentra correctamente[6].

La información que se muestra es la siguiente:

- *Product ID*: Es el identificador del producto.
- *Sub Org*: Se refiere al cliente asociado al producto.
- *Unit Value*: El valor del precio del producto en la tabla TSI_PD_V_PRICE, esta tabla almacena los precios de los productos.
- *Currency*: Indica el tipo de moneda asociado.
- *Ship From*: Se refiere al proveedor del producto.
- *Ship To*: Es el destinatario del producto.
- *Created By*: Se refiere al usuario o sistema que creó el registro en TSI_PD_V_PRICE.
- *Created Date*: Es la fecha en que el registro fue creado.
- *Last Modified By*: Se refiere al usuario o sistema que modificó el registro por última vez.
- *Last Modified Date*: Es la fecha en que el registro fue modificado por última vez.

Al oprimir el botón *Save*, aparecerán inmediatamente las modificaciones de los campos *Last Modified By* y *Last Modified*

[6] El precio tiene la restricción de que debe ser un número. Los demás campos no podrán ser modificados.

Date por el usuario y la fecha actual, respectivamente. A continuación, se muestran los estados de pantalla que representan esta sección. Como se observa en la Figura 7 y la Figura 8 el valor de *Last Modified Date* ha sido actualizado automáticamente después de cambiar los valores de *Unit Value* y *Currency* y oprimir el botón *Save*.

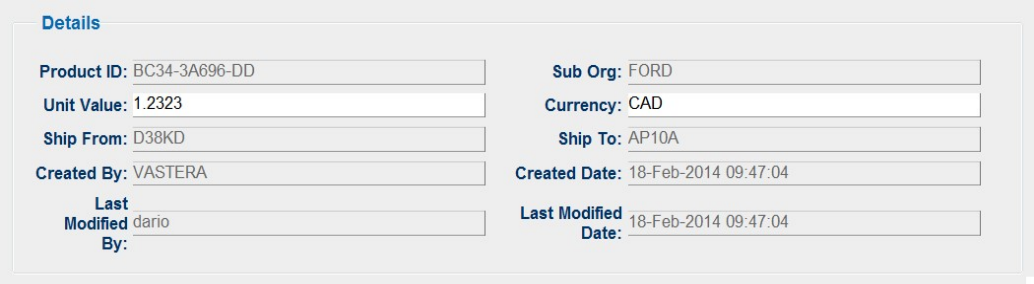

Figura 7. Detalles del producto antes de guardar.

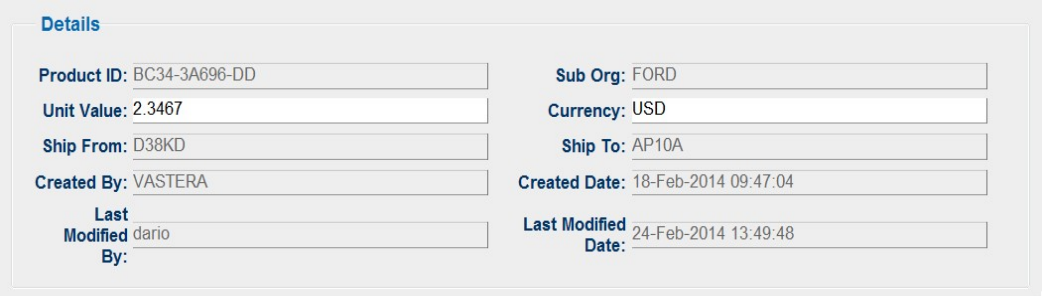

Figura 8. Detalles del producto después de guardar.

Ventana emergente

Al oprimir alguno de los botones de búsqueda, aparecerá una ventana emergente que le permitirá al usuario buscar los registros que necesita consultar o modificar. Está compuesta por dos secciones, la primera, muestra las opciones de

búsqueda que son los filtros que definen los criterios para encontrar el producto que se está buscando.

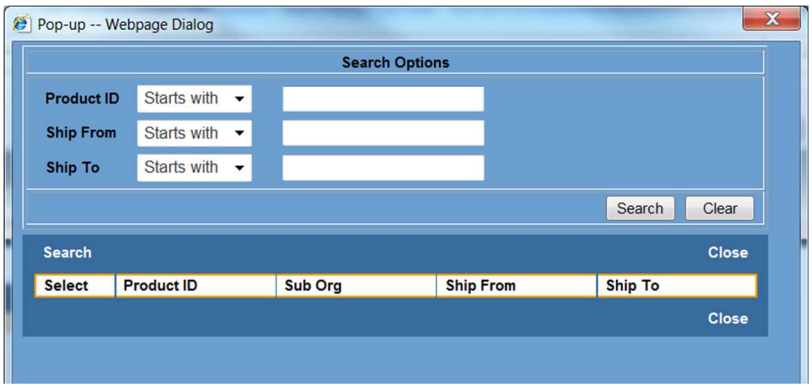

Figura 9. Pantalla de la ventana emergente para buscar productos.

Opciones de búsqueda

En esta sección de la pantalla, el usuario final podrá especificar las condiciones que se deben cumplir para filtrar los registros a consultar y/o modificar. El contenido de la caja de texto se comparará con el valor en la base de datos respectivo usando el criterio de emparejamiento seleccionado en el menú desplegable. Para todos los filtros, los criterios de emparejamiento son:

- *Starts with* (Comienza con): Acepta si el valor contiene al inicio la cadena especificada en la caja de texto.
- *Contains* (Contiene): Acepta si el valor contiene la cadena en la caja de texto sin importar la ubicación.
- *Is Equal to* (Es igual a): Acepta si el valor es exactamente igual a la cadena en la caja de texto.

Dependiendo del origen de la ventana emergente que se eligió en la sección de búsqueda de productos de la pantalla

principal, la ventana emergente cambia los criterios de filtro:

- Búsqueda por producto: Esta opción tiene tres criterios de búsqueda: Identificador de producto, proveedor y destinatario. El usuario podrá buscar filtrando los productos que cumplan el criterio elegido. En la Figura 10 se observan los tres campos de la base de datos con los que se comparan los filtros de búsqueda.

Figura 10. Opciones de filtro de la ventana emergente que aparecen al apretar el botón de búsqueda de producto.

- Búsqueda por identificador de producto: Esta opción busca un producto usando solamente el criterio de identificador de producto. En la Figura 11 se observa el criterio de búsqueda asociado solamente al identificador del producto.

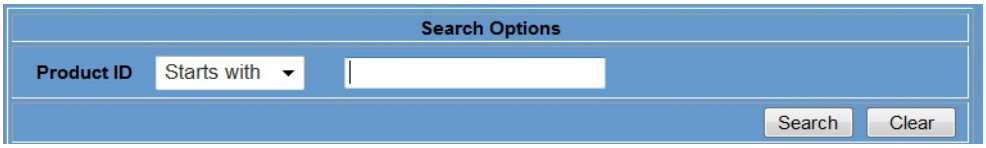

Figura 11. Opción de filtro de la ventana emergente que aparece al apretar el botón de búsqueda asociado al identificador del producto.

- Búsqueda por proveedor: Esta opción busca un producto usando solamente el criterio de proveedor. En la Figura 12 se observa el criterio de búsqueda asociado solamente al proveedor.

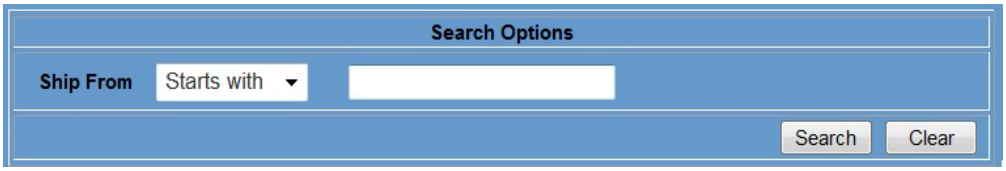

Figura 12. Opción de filtro de la ventana emergente que aparece al apretar el botón de búsqueda asociado al proveedor.

- Búsqueda por destinatario: Esta opción busca un producto usando solamente el criterio de destinatario. En la Figura 13 se observa el criterio de búsqueda asociado solamente al destinatario.

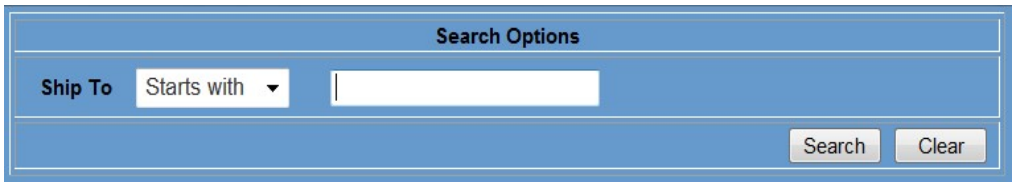

Figura 13. Opción de filtro de la ventana emergente que aparece al apretar el botón de búsqueda asociado al destinatario.

Si se oprime el botón *Search* se inicia el proceso de búsqueda donde la plataforma considerará sólo los campos que no estén vacíos. Oprimir el botón *Clear* reestablece los campos de esta sección.

Si no se elige un criterio, una ventana aparecerá para confirmar al usuario que no hay filtros para la información y sólo los primeros 50 registros se van a mostrar. En la Figura

14 se observa la ventana emergente de confirmación cuando todas las cajas de texto de los filtros se encuentran vacías.

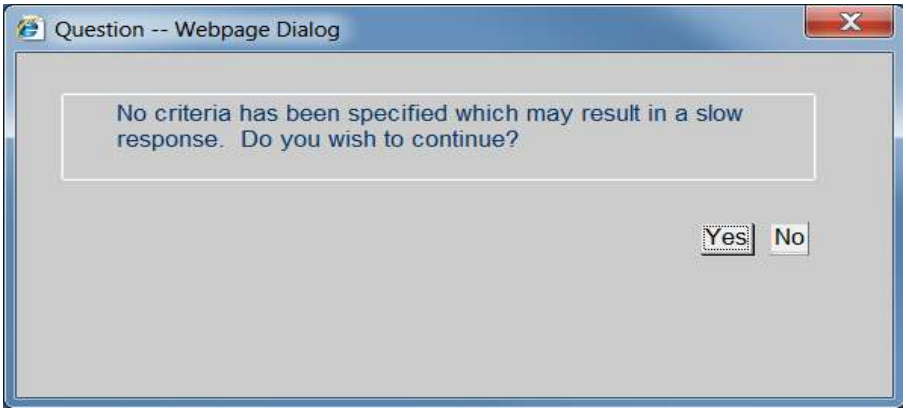

Figura 14. Ventana emergente para confirmar cuando no hay filtros para la búsqueda.

Resultados de búsqueda

Después de que el usuario final presione el botón *Search* de las opciones de búsqueda, se mostrarán los registros que cumplan con los criterios de filtros especificados[7]. La Figura 15 muestra un resultado de búsqueda por medio de la opción de búsqueda por destinatario. Se observa que tres productos cumplieron con el criterio de búsqueda.

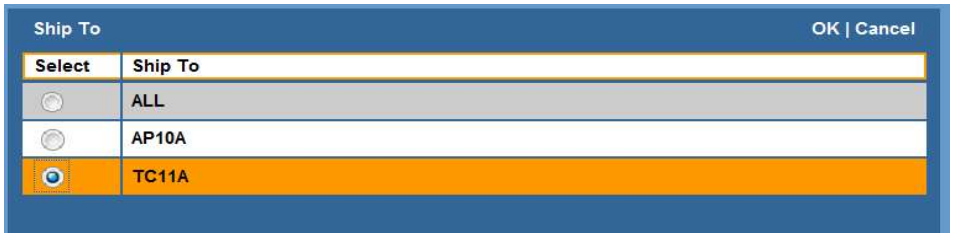

Figura 15. Pantalla con los resultados de una búsqueda.

[7] El número máximo de elementos que puede mostrar es 50. Entre más específica sea la búsqueda, más sencillo será encontrar el registro deseado.

Si el usuario final oprime el botón *OK*, la ventana emergente se cerrará y mostrará en la pantalla principal el valor elegido en el campo de origen de la búsqueda. Oprimir el botón *Cancel* cerrará la ventana emergente sin consecuencias en la pantalla principal.

La búsqueda por producto mostrará el resultado permitiendo sólo elegir un resultado debido a que esta opción está diseñada para encontrar un solo producto. Las demás búsquedas mostrarán los resultados permitiendo seleccionar más de un resultado.

Capítulo 3. Implementación del proyecto MultipriceCopyRule

El proyecto de *Transaction Automation* (TA) es un modelo **prototipado** el cual, conforme las necesidades del negocio lo exigían, se le agregaron más funcionalidades, como la regla *Multiprice* que detenía el proceso al encontrar productos con varios precios.

Un modelo *prototipado* es útil con proyectos grandes para los cuales no se tiene una idea clara del alcance del producto final por parte de los dueños del proyecto. Este modelo permite que todo el sistema, o algunos de sus partes, se construyan rápidamente para comprender con facilidad y aclarar ciertos aspectos en los que se aseguren que el desarrollador, el usuario y el cliente estén de acuerdo en lo que se necesita, así como también la solución que se propone para dicha necesidad y de esta forma minimizar el riesgo y la incertidumbre en el desarrollo[8].

La implementación del proyecto *Multiprice* es una iteración que surgió a raíz de la necesidad de reducir costos de operación. El equipo operativo descubrió una lógica dentro del negocio que reduciría la cantidad de productos donde se requería una intervención manual para obtener el precio correcto, a raíz de esto, generaron el requerimiento para implementar esta nueva lógica.

La implementación del proyecto fue realizada completamente por el autor de este documento, siendo el primer proyecto en

[8] Fuente:Ingeniería de software. Por Shari Lawrence Peleeger.

la compañía realizado por él. Durante la ejecución del proyecto se ignoraban procesos (los equipos de operación, UAT y desarrollo no los seguían), lo cual llevó a varias dificultades que retrasaron su finalización. Estas dificultades fueron ocasionadas por las siguientes situaciones:

- Cambios provenientes del cliente a la mitad del proyecto.
- Mal dimensionamiento sobre el impacto que producirían los cambios.
- Inconsistencias de la Plataforma Colaborativa (En inglés *Collaborative Plattform*, CP).

Cambios provenientes del cliente

A continuación se describen los cambios provenientes del cliente. Estos cambios ocasionaron repetir partes del desarrollo y retraso del proyecto.

El cliente no tuvo claros los requerimientos del negocio, no consideraba posibles escenarios que se podían presentar. Algunos escenarios fueron omitidos porque no se le ocurrieron y otros porque creían que era imposible que ocurrieran. De igual forma, se tuvieron que revisar constantemente todos los escenarios posibles y el cliente tenía que determinar la forma en que se debía manejar cada uno de ellos. Por este medio llegó la solución de considerar que los productos en la tabla de `TSI_PD_V_PRICE` tuvieran un valor de proveedor y destinatario como `'ALL'` para los productos que no tenían solo un valor determinado para ese campo en la base de datos.

De igual forma, se agregaron todas las distintas combinaciones de valores de las variables y cómo manejarlas.

Dimensionamiento incorrecto de los impactos del cambio

Una de las principales situaciones que ocasionan la repetición de trabajo en el desarrollo de nuevos módulos o correcciones de los actuales es cuando no se dimensionan los posibles impactos que puede tener el cambio a implementar. *TreadeSphere Importer* (TSI) y *Transaction Automation* (TA) son desarrollos muy complejos y cada elemento genera variaciones que pueden alterar la consistencia de datos si no son manejados adecuadamente. Al cambiar el funcionamiento de algún elemento, puede alterar el de los demás.

Cálculo incorrecto de los precios totales

La regla *MultipriceCopyRule* se desarrolló para identificar el precio correcto de los productos que tienen varios precios, sustituyendo a la regla *Multiprice*. En un inicio se colocó la regla *MultiPriceCopyRule* en el mismo orden que la regla anterior. En las pruebas de *User Acceptance Testing* (UAT), que son las encargadas de asegurar que la solución es adecuada para las necesidades del cliente, se identificó que los totales de los precios no se calculaban correctamente. Esto era realizado por la regla *CalculationValue*, la cual toma el precio del producto y lo multiplica por la cantidad de unidades del producto. La información del producto se conseguía con la regla de obtención del producto (*ProductSubPullRule*) que se ejecutaba al inicio del proceso y encuentra el último precio en el historial. El cálculo del total se realizaba justo después de la regla posterior a la obtención del producto (*PostProductSubpullRule*) y la regla

MultiPriceCopyRule se ejecutaba al final. Con este orden, se efectuaba el cálculo antes de encontrar el precio correcto y en los casos en que la regla *MultipriceCopyRule* encontraba uno distinto al encontrado inicialmente, las cantidades no coincidían.

Este problema se resolvió alterando el orden de las reglas de tal manera de que la regla que obtiene el precio correcto cuando un producto tiene precios múltiples (*MultipriceCopyRule*) se ejecuta justo después de que la regla posterior a la obtención del producto (*PostProductSubPullRule*) termina de ejecutarse. En la Figura 16 se muestra la comparación del orden anterior de las reglas y el orden final que soluciona el problema mencionado en este apartado.

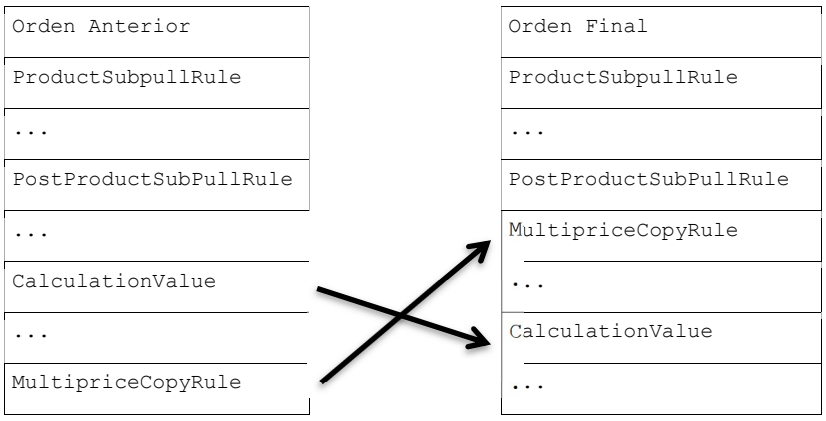

Figura 16. Cambio del orden de las reglas.

Error en la regla posterior a la obtención de producto (*PostProducSubpullRule*)

Un error en la lógica de la regla posterior a la obtención de producto (*PostProductSubpullRule*) se desenmascaró al quitar la lógica original de detección de productos con precios múltiples.

Una vez terminado el proyecto, dentro de las pruebas de aceptación del usuario (UAT), se encontraron casos que enviaban un mensaje de error por no tener precio, que anteriormente se enviaba como error por ser un producto con múltiples precios. Este error detenía la ejecución sin darle oportunidad a la regla que obtiene el precio correcto cuando un producto tiene precios múltiples (*MultipriceCopyRule*) de obtenerlo. Este mensaje ocurría en la regla posterior a la obtención de producto (*PostProductSubPullRule*), la cual se tenía que ejecutar antes. Si el proceso se ejecutaba por segunda vez, terminaba exitosamente, pero el objetivo del proyecto era evitar la intervención manual lo más posible.

Para esto se modificó la regla posterior a la obtención del producto (*PostProductSubPullRule*) para que en lugar de enviar un mensaje de error y detener la ejecución, sólo se envíe un mensaje de advertencia y continuara con la ejecución.

Inconsistencias de la Plataforma Colaborativa (CP)

Durante el desarrollo de la interfaz gráfica, se identificó que los registros modificados manualmente no se estaban guardando correctamente en la base de datos. La identificación de este problema retrasó el proyecto una semana.

La aplicación web de *TradeSphere Importer* (TSI) utiliza un desarrollo propio llamado Plataforma Colaborativa (CP) la cual es un desarrollo compuesto por Javascript (capa de presentación), Java (capa de acceso de datos) y con conexión a la base de datos Oracle. En la implementación de la función de almacenamiento de datos (*Save*), la entidad de datos no conservaba los valores correctos. La depuración de la Plataforma Colaborativa llevó varios días y se encontraron

procesos intermedios fuera del alcance del explorador que se realizaban en la capa de acceso de datos.

La decisión tomada fue reemplazar esa funcionalidad y llenar la entidad de datos ignorando la biblioteca de la Plataforma Colaborativa y preparando la información para que la Plataforma Colaborativa retomara el control para que la interfaz continuara funcionando de manera normal.

Se utilizaron las herramientas del explorador *Chrome* para depurar el código de Javascript y así encontrar el problema (el cual no se podía modificar ya que tenía que ser un cambio directo a la Plataforma Colaborativa y podía tener un impacto significativo a las demás aplicaciones). De esta forma se puedo comprender el funcionamiento de la Plataforma Colaborativa para preparar la información de la forma requerida.

Una problemática importante en este tema es que la Plataforma colaborativa no estaba debidamente documentada, esto hubiera reducido el tiempo requerido para identificar y corregir el problema.

Capítulo 4. Resultados obtenidos

El proyecto de la regla de *MultipriceCopyRule* terminó exitosamente después de ocho ciclos de desarrollo. La regla pasó las pruebas de aceptación del usuario (UAT) llegando al ambiente de Producción.

Esta regla reduce el tiempo que un analista dedica al proceso de validación de las facturas obteniendo automáticamente el precio correcto de un producto de precio múltiple. Anteriormente un analista cambiaba el precio manualmente en el sistema cada vez que un producto de precio múltiple era analizado. Cabe resaltar que con la nueva regla implementada, el analista sólo cambiará manualmente el precio del producto cuando no sea posible encontrar el precio automáticamente.

Además la lógica de esta regla recuerda la última interacción del usuario evitando que tenga que intervenir manualmente para el mismo producto con el mismo valor de proveedor y destino.

Las situaciones presentadas en el Capítulo 3 obligaron a la empresa a retomar un proceso riguroso para las peticiones de cambios que reduzca el tiempo y los ciclos de desarrollo.

- Al recibir una petición de cambio se realizará una junta para analizar las especificaciones. Esta junta involucrará a los expertos en la lógica del negocio para que puedan detectar los posibles impactos que los cambios puedan tener en el resto de la aplicaciones y los distintos escenarios que se deben contemplar durante la implementación del cambio.

- El desarrollador deberá generar un documento con la solución a implementar que se incluirá en la petición del cambio, de esta forma, el cliente aprobará o rechazará el cambio detallado, delimitando en un inicio los alcances de la implementación.
- El documento debe ser aprobado por quien solicitó la petición de cambio, quien debe proveer un plan de pruebas unitarias. De igual forma, deberá ser avalado por los expertos en la aplicación. Esto obliga al solicitante a crear otra petición de cambio en el caso de que quiera modificar los alcances.
- Al terminar la solución se debe hacer una revisión de código con los compañeros de trabajo que tengan mayor conocimiento sobre la plataforma. Así revisar si la implementación se hizo de acuerdo a las mejores prácticas y asegurar que cumpla correctamente antes de entregar al cliente.
- Al concluir el proceso de pruebas, se hace una transferencia de conocimiento y se muestra la funcionalidad para formalizar la entrega al cliente.
- Después de ser aprobado el cambio, el código se empaqueta con su archivo de instrucciones (*README*) que debe contener las instrucciones detalladas para instalar y desinstalar la actualización.

La finalidad de las adecuaciones al proceso es reducir los ciclos de revisión, disminuir los cambios al código y las pruebas de aseguramiento de calidad (QA) y aceptación del usuario (UAT). También limita al inicio el alcance del cambio para que el área de desarrollo pueda cobrar el cambio y no estar en una actualización constante e interminable. Ya que

en la implementación que describe este documento, los ocho ciclos de desarrollo fueron cobrados como un cambio, a pesar de que el cliente no consideró el requerimiento debidamente antes de solicitarlo.

Lo anterior ha generado un impacto positivo en los proyectos posteriores reduciendo el tiempo de entrega y recursos utilizados para validar los proyectos.

A partir de este proyecto, las demás peticiones de cambios se cerraron a lo más en tres ciclos de desarrollo.

Conclusiones

El proyecto de la regla *Multiprice*, aunque problemático, se desarrolló de manera exitosa, ya que se redujo tiempo y dinero a la empresa al requerir menor intervención manual por parte del área de operaciones. Generó retos profesionales desde sobrellevar y entender los problemas de logística de la empresa hasta entender el código de proyectos realizados por distintas personas hace muchos años (la última actualización de las reglas fue tres años previos al proyecto), quienes ya no estaban en la compañía para asesorar. El código de la Plataforma Colaborativa fue creado en la década de los noventas y aunque sea robusto, identificar los errores que tiene es complicado. El código tenía que adaptarse a la Plataforma Colaborativa corrigiendo los errores producidos por la Plataforma Colaborativa e integrando dichas correcciones con el resto de los requisitos del desarrollo.

Se mejoró la relación con las distintas áreas de la compañía al solucionar exitosamente un problema considerado complejo por ellos y de este modo la confianza en los proyectos subsecuentes.

Durante el desarrollo de este proyecto:

- Se aprendieron métodos para depurar minuciosamente los procesos de *Javascript* cuando no se tiene acceso o permiso para modificar el código del mismo y entender el funcionamiento de la Plataforma Colaborativa.

- Se aprendió una forma de diseño de base de datos jerárquica que permite un manejo sencillo de las entidades contenidas en otras entidades.
- Se corroboró la importancia de la documentación, tanto la problemática de la falta de documentación de la Plataforma Colaborativa (el *Javadoc* existe, pero no explicaba nada, sólo declaraba variables y funciones) así como la necesidad de documentar el trabajo actual tanto para definir un buen diseño como para delimitar las obligaciones y no trabajar doble sin que el cliente pague.
- Se aprendió a utilizar archivos XML para facilitar la programación de procesos web. *TradeSphere* utiliza funciones de *Javascript* que leen archivos XML y a partir de eso, generan el código de elementos como las ventanas emergentes.

Gracias a la materia de Ingeniería de Software fue posible corregir los problemas de protocolo y generar una documentación del proyecto. De esta y otras experiencias profesionales, buscaría una forma de aterrizar los conceptos enseñados en esta materia. Donde se pueda comparar el proceso deseable con situaciones que suceden en compañías pequeñas, medianas y grandes. De esta forma, los estudiantes podrán tener una mejor perspectiva de los problemas que se enfrentarán en su trabajo y anticiparse a la identificación y corrección de los procesos mal diseñados o mal ejecutados.

Las materias de Introducción a las Ciencias de la Computación y Lenguajes de programación ayudaron con las habilidades necesarias para programar la regla *Multiprice* y analizar el código existente. Con la experiencia adquirida durante mi

trabajo, hay conceptos que introduciría en estas materias, o bien, agregaría una materia que vea el proceso de arquitectura de software. Esto está pensado con el motivo de ayudar a idear soluciones complejas considerando conectividad entre componentes, tecnologías a usar y código extensible basado en conceptos de buenas prácticas en el diseño del software para cuando se trabaja con distintos equipos o, como en el caso de este proyecto, con desarrollos existentes donde no se cuenta con el equipo que desarrolló el código original.

La materia de Base de Datos permitió entrar a la empresa ya que fue parte importante del examen de admisión ya que el manejo de los datos para obtener la información necesitada fue crucial para la funcionalidad adecuada. Esta materia dio una buena introducción, pero no tomó temas relevantes de complejidad como: situaciones en las que conviene usar procedimientos almacenados o el orden ideal de unir consultas para una mejor eficiencia en la búsqueda de información.

Bibliografía

1) Varios. (2005). *TradeSphere InfrastructureTM API Specification*. 6/08/2015, del repositorio de Livingston International Sitio: file:///%doc_root%/TradesphereDoc/Javadoc/VasteraTools/index.html

2) Darío Urdapilleta. (2014). *Copy Rule Improvement For Transaction Automation System Design Specifications*. 28/05/2015, del repositorio de Livingston International: FORD - TX Automation - CRI Analysis - Multi-Pricing Copy Rule - SDS v0.5.2.doc

3) Livingston International. (2014). *History – Customs brokerage, freight, and trade compliance service professionals*. 17/05/2015, de Livingston International Sitio web: http://www.livingstonintl.com/about-us/history/

4) Gamma E., Helm R., Johnson R. y Vlissides J., *Design Paterns: Elements pf Reusable Object-Oriented Software*. Reading, MA, USA: Addison-Weley Professional, 1994.

5) McConnell S. (1996). *Rapid Development: Taming Wild Software Schedules*. Redmond, Washington, USA. Microsoft Press.

6) Sommerville I. (2005). *Software Engineering*. Novena Edición. Boston, MA, USA. Pearson.

7) Freeman S. y Pryce N,. (2009). *Growing Object-Oriented Software, Guided by Tests*. Primera Edición. Boston, MA, USA. Pearson.

Anexo

A continuación se explica el proceso ilustrado en la Figura 17.

La solución tiene como datos de entrada el cliente, el código de país, el identificador de la factura y el número de versión. Realiza lo siguiente:

1) Obtiene de la base de datos la lista de facturas y productos que coincidan con los parámetros de entrada.
2) Por cada uno de los productos en cada factura:
3) Se realiza el subproceso *checkMultipriceForProduct()* descrito en la Figura 18 el cual obtiene el valor de la bandera de precios múltiples.
4) Si el proveedor y destinatario son vacíos y la bandera de precios múltiples no es 'N' envía el mensaje 9728 y se terminar el proceso con error. Esto significa que el producto no tiene bandera de precios múltiples y tampoco tiene destinatario o proveedor. En la lógica del negocio, alguno de estos valores debe existir.
5) Si tiene destinatario y proveedor, obtiene el valor del precio, identificador del proveedor y fecha con la fecha más reciente.
    1. Si no encuentra el precio o la bandera de precios múltiples es 'N', se asigna el destinatario y proveedor como 'ALL'. Esto significa que no es un producto con precios múltiples.

2. Se obtiene de la base de datos el valor del precio, el identificador y fecha con la fecha más reciente sin considerar el destinatario y proveedor.

6) Se verifica si el identificador del proveedor obtenido es igual a -1. En caso de no serlo, se realiza el subproceso *getVIDFromProduct()* descrito en la Figura 19 que obtiene el valor del identificador.

   1. En caso de no poder obtener el valor del identificador del proveedor usando el subproceso *getVIDFromProduct()* se envía el mensaje 9729 que indica que no se encuentra un proveedor para el producto. Se termina el proceso con error.

7) Teniendo el identificador del proveedor se obtiene la información del precio de la nueva tabla de precios.

   1. Si el precio no existe o su valor es vacío, se inserta en la nueva tabla de precios.

8) Si el valor del precio es igual a cero tanto en el historial de los precios como en la nueva tabla de precios, se envía el mensaje 9729 y se termina el proceso con error, debido a que el precio no puede ser cero.

9) Si ambos existen y el valor en el historial e más reciente que en la nueva tabla de precios, o si el valor en la tabla de precios es cero y en el historial es distinto de cero, entonces se actualiza la nueva tabla de precios para tener el precio correcto más reciente.

10) Una vez asegurado que se tiene un precio válido que está actualizado en la nueva tabla de precios, se actualiza el precio en la factura y se termina el proceso con éxito para ese producto de esa factura.

11) Esto termina el ciclo iniciado en el paso 2.

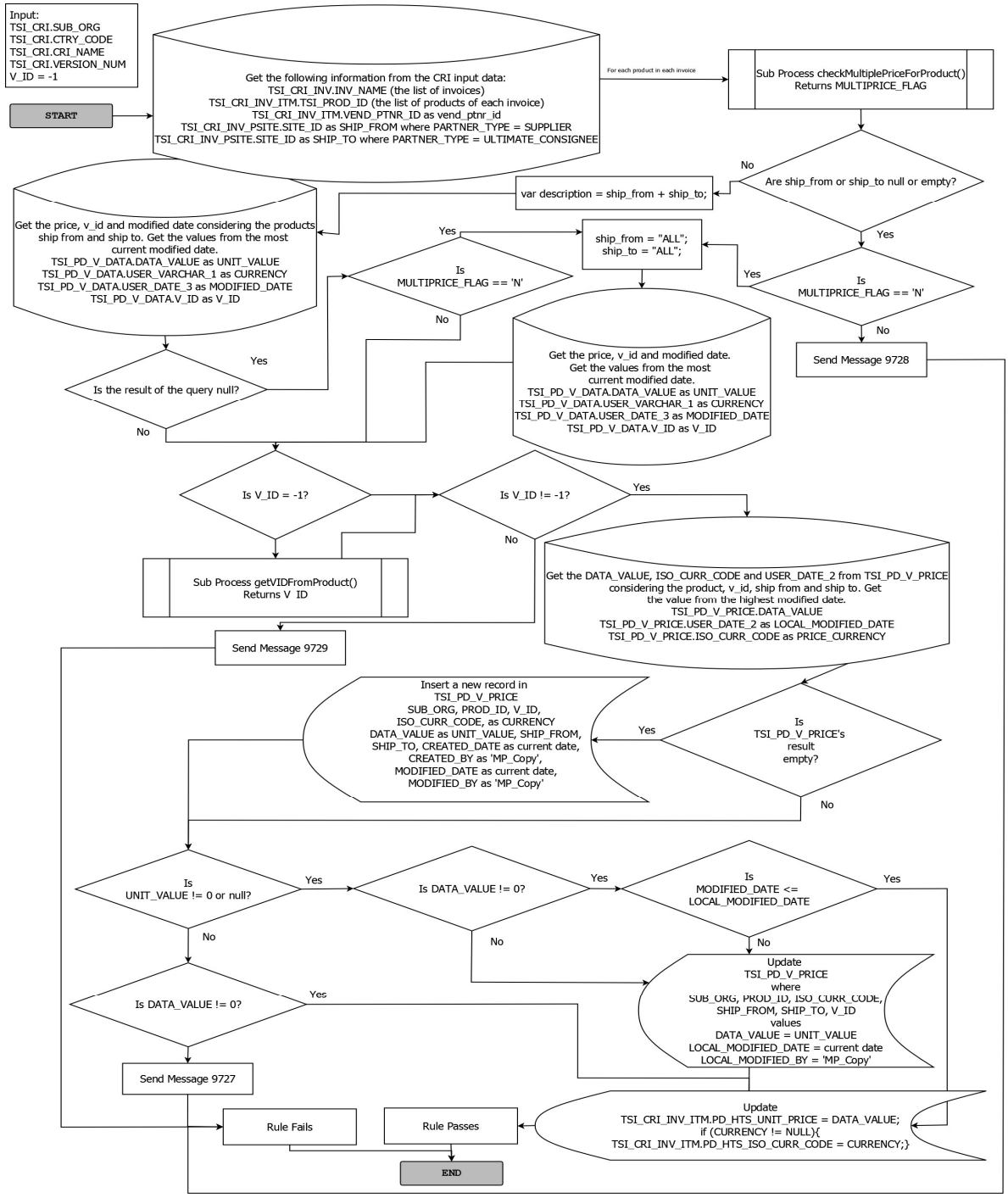

Figura 17. Proceso lógico de la nueva regla *Multiprice*.

A continuación, se describe el subproceso *checkMultiPriceForProduct()* ilustrado en la Figura 18.
Los datos de entrada son el cliente y el identificador del producto en la factura.

1) Se obtiene la variable donde se guarda si el producto puede o no tener varios precios de la tabla de productos, a esto le conoce como la bandera de precios múltiples.
2) Se regresa el valor de la bandera de precios múltiples.

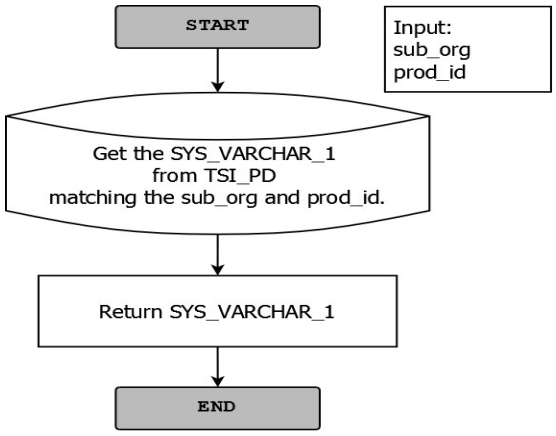

Figura 18. Ilustración del subproceso *checkMultiPriceForProduct()*.

A continuación, se describe el subproceso *getVIDFromProduct()* ilustrado en la Figura 18.
Los datos de entrada son el identificador del producto, el proveedor, destinatario y bandera de precios múltiples. Este subproceso asume que el identificador del proveedor no se encuentra en el historial.

1) Se obtiene de la tabla de proveedores el identificador que coincida con los valores de entrada.
    1. En caso de ser nulo o vacío y la bandera de precios múltiples es 'N', es decir que es un producto que no tiene precios múltiples, se busca de nuevo solamente

coincidiendo el identificador del producto y el cliente.

2) Se regresa el valor del identificador del vendedor obtenido.

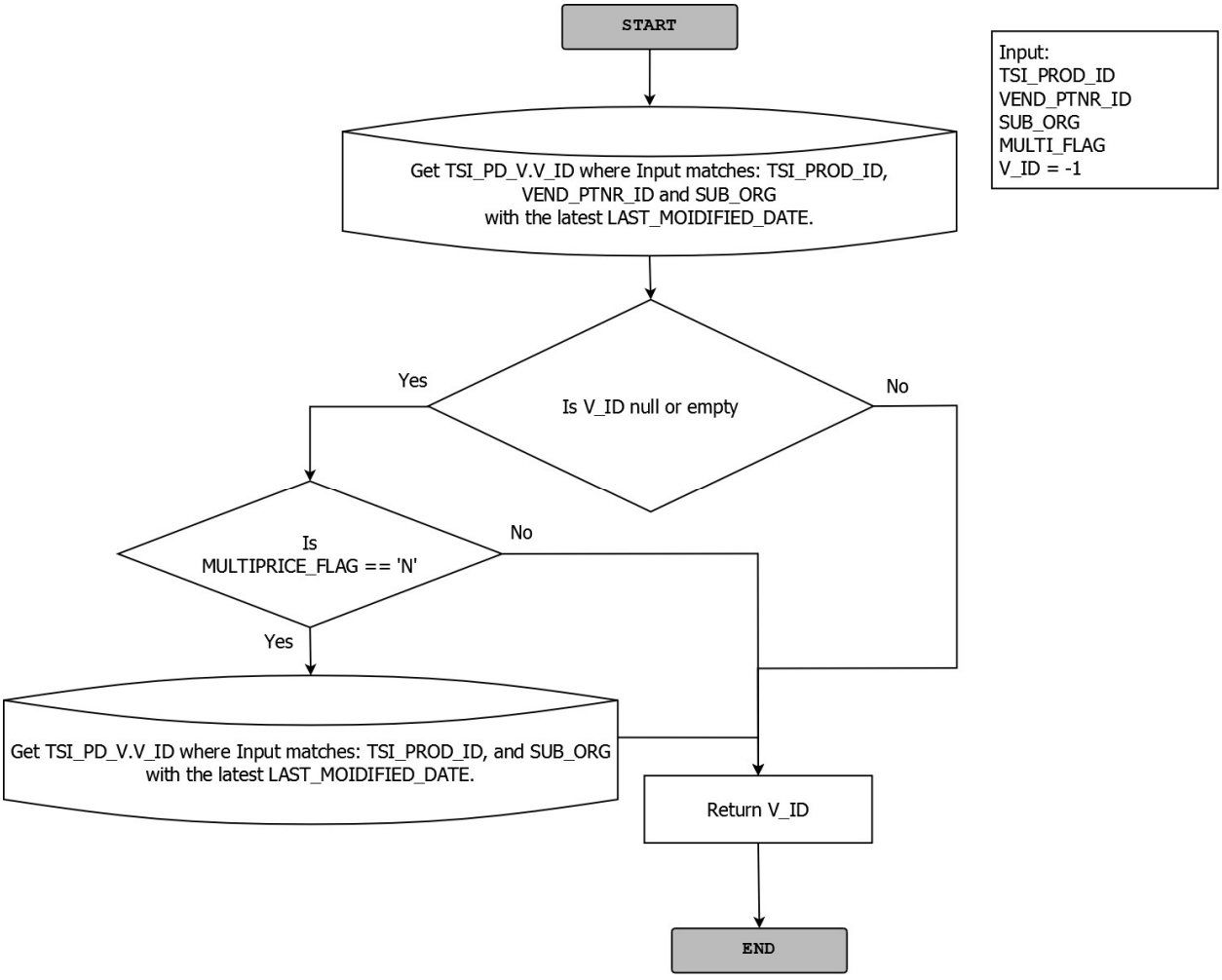

Figura 19. Ilustración del subproceso *getVIDFromProduct()*