Universidad de las Ciencias Informáticas

Facultad 3

Herramienta para comparar procesos de negocio usando la técnica de Cubos de Procesos

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores: Leandro Luis Rojo

Omar Sablón Mirabal

Tutores: Ing. Dina Torres Sakipova

MSc. Damián Prez Alfonso

La Habana, 2015, “Año 57 de La Revolución”

DECLARACIÓN JURADA DE AUTORÍA

Declaramos ser los autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los __ días del mes de ___del año ___.

Leandro Luis Rojo

_____

Firma del autor

Omar Sablón Mirabal

_____

Firma del autor

Ing. Dina Torres Sakipova

_____

Firma de la tutora

MSc. Damián Pérez Alfonso

_____

Firma del tutor

DATOS DE CONTACTO

Tutora:

Nombre y apellidos: Ing. Dina Yaksilik Torres Sakipova

Correo electrónico: dytorres@uci.cu

Situación laboral: Profesor

Institución: Universidad de las Ciencias Informáticas (UCI).

Dirección: Carretera a San Antonio de los Baños, Km 2 1/2, Reparto Torrens, Boyeros.

Tutor:

Nombre y apellidos: MSc. Damián Pérez Alfonso

Correo electrónico: dalfonso@uci.cu

Situación laboral: Profesor

Institución: Universidad de las Ciencias Informáticas (UCI).

Dirección: Carretera a San Antonio de los Baños, Km 2 1/2, Reparto Torrens, Boyeros.

AGRADECIMIENTOS

De Leandro:

A mis familiares que me han apoyado en todo.

A mi compañero de tesis que se complementó en tantos malos momentos.

Al profesor Damián que sin él no hubiese podido existir este trabajo de diploma.

A nuestra tutora Dina.

A mi novia que la quiere tanto.

A mis compañeros de la carrera, en especial a René y Leosbel.

De Omar:

A mis familiares que me han apoyado en todo. En especial a mis padres y mis hermanas que han pasado por tanto en los últimos tiempos y han sabido sobreponerse.

A mi compañero de tesis por ser mi hermano de mil batallas.

A nuestra inquieta tutora por sus regaños

A Damián simplemente por existir y por estar dispuesto a ayudarnos sin pedir nada a cambio, y por ser un ejemplo de ahínco y consistencia en cuanto conocimiento se refiere.

A mis amigos de la UCI sin los cuáles mi vida hubiera sido diferente.

A mis compañeros de aula por soportarme durante cinco años

A mis amigos del barrio por anhelar este momento tanto como yo, en especial a Yosvany, Henry y Jorge Cesar

A los profesores que me hicieron un profesional

A los que de una manera u otra me apoyaron durante la carrera

DEDICATORIA

De Leandro:

A mi hermano que sin él no hubiese podido terminar la carrera.

A mis padres para que estén orgullosos.

De Omar:

A mis abuelos Eloy y Onelia que ya no están aquí, pero que siempre fueron mi ejemplo a seguir y gracias a ellos soy quien soy con defectos y virtudes.

A mis padres por enseñarme a ser mejor y por sobreponerse a las adversidades

A mis hermanas por apoyarme en todas mis decisiones y ayudarme en todos mis problemas.

RESUMEN

Recientemente ha existido un creciente interés por comparar múltiples procesos a la vez en búsqueda de semejanzas o diferencias que favorezcan ventajas competitivas. Este tipo de análisis corresponde a la fase de Diagnóstico dentro de la Gestión de Procesos de Negocio, y solo una buena aplicación de la misma permite el éxito de cualquier negocio. La Minería de Procesos es un área útil en este sentido, pues está enfocada en los procesos y a la vez favorece el Diagnóstico de Procesos, pero tiene limitaciones que obstaculizan los análisis comparativos. Los Cubos de Procesos son una herramienta que mitigan en gran medida estas limitaciones por lo que el objetivo planteado para el presente trabajo fue crear un Cubo de Procesos que permita obtener información teniendo en cuenta diferentes características y utilizar esta información para compararla con otra resultante de otros análisis similares, además que divida una gran cantidad de información en pequeñas partes para hacer más fácil el análisis. La solución propuesta constituye un Complemento para el marco de trabajo ProM y es una mejora de la versión experimental de Tatiana Mamaliga (Mamaliga, 2013), de forma que elimina las limitaciones que tenía esta y añada otros aspectos importantes como la comparación a nivel de datos. Al realizar este trabajo se puede concluir que el enfoque de los Cubos OLAP aplicados a la Minería de Procesos es una vía efectiva para mitigar las limitaciones en el análisis de los procesos.

Palabras claves: análisis, comparación, Cubos de Procesos, diagnóstico, Minería de Procesos.

ABSTRACT

In the last times there is an increasing interest for comparing some processes at the same time in order to find similarities and differences that allows competitive advantages. This kind of analysis belongs to Diagnostics phase into Business Process Management and only by a proper applying of itself allows to organizations have success.

Process Mining is a useful area focused on processes and at the same time helps to Diagnostics phase; however it has limitations that hinder comparatives analysis. Process Cube is a tool that mitigates in a great manner these limitations, for that reason the aim of present work was to develop a Process Cube to obtain information having into account different characteristics, and then use this information for comparing them with other similarities analysis results. Besides, Process Cube is based in “divide and conquers” approach; therefore it divides a great amount of information in smaller parts in order to improve the analysis. The proposed solution is a Plugin from ProM framework and improves the experimental version presented by Tatiana Mamaliga so that removes the limitations from the older version and adds new functionalities like compare by numerical data. In conclusion, the OLAP approach joined to Process Mining is an effective way to mitigate the problems in Process Diagnostics.

Keywords: analysis, comparing, Diagnostic, Process Cube, Process Mining.

ÍNDICE

INTRODUCCIÓN

La gestión de los procesos de negocio (*Business Process Management*, BPM) está reconocida como un enfoque que promueve la efectividad en los negocios y la eficiencia, a la vez que se empeña por la innovación, flexibilidad e integración con la tecnología. BPM se puede definir como el arte y ciencia de supervisar cómo el trabajo es realizado en una organización para asegurar resultados consistentes y tomar ventaja para mejorar oportunidades (Sadiq, y otros, 2014). Recientemente se ha dado a conocer una nueva variante del BPM, denominada iBPM (Daniel, y otros, 2013) o dicho de otra forma BPM inteligente, la cual ha dado un nuevo ímpetu a la integración de tecnologías analíticas para la organización de procesos. Esto está permitiendo a organizaciones líderes diseñar sus operaciones de negocio de una forma más inteligente y dar a los participantes en los procesos una mejor perspectiva situacional en tiempo real, además de propiciar la habilidad para ajustar sus responsabilidades apropiadamente.

Una de las etapas dentro de la Gestión de Procesos de Negocio es el Diagnóstico de Procesos, que es, en definitiva, donde se dictan las acciones a realizar en dependencia de un determinado resultado o comportamiento de los procesos. El Diagnóstico de Procesos se puede definir como la fase del BPM que evalúa los procesos y monitorea requerimientos emergentes debido a cambios en el entorno de los procesos (van der Aalst, 2011). Una de las disciplinas que facilita el desarrollo de esta fase es la Minería de Procesos (MP), la cual parte de un registro de eventos para realizar todo su análisis. El desafío está en aprovechar los datos de eventos en una forma significativa, por ejemplo, para proveer un mejor entendimiento, identificar cuellos de botella, anticipar problemas, registrar violaciones de políticas, y simplificar procesos. Dentro de las potencialidades que tiene no solo se halla la posibilidad de realizar el diagnóstico, sino que también, mediante los datos de eventos guardados por los sistemas informáticos, aplica algoritmos de descubrimiento y mejora de los modelos de procesos, o brinda la posibilidad de chequear si la ejecución de un proceso marcha según lo esperado (van der Aalst, 2011).

Estas potencialidades hacen que cada vez más las organizaciones se interesen por el desarrollo de técnicas de MP en su entorno para favorecer ventajas competitivas. Un ejemplo de esto es el departamento de control del tráfico de Italia, encargado de velar por el cumplimiento de las leyes en la vía y por lo tanto es el encargado de generar las multas en caso de violaciones. En este departamento se generan a diario cientos o miles de multas, reclamaciones y retrasos en el pago. El tiempo de respuesta a las reclamaciones es elevado y el análisis del proceso no aporta evidencia suficiente para tomar decisiones. Además, es necesario detectar problemas en el proceso, saber quién está cometiendo errores o en qué parte del proceso se están cometiendo. Por otra parte, es probable que se produzcan aumentos de multas en determinadas épocas del año o que la cantidad y el monto de las multas varíen de una región a otra. Para conocer este tipo de información útil, es necesario dividir el proceso según diferentes criterios, la mayoría de los Complementos no permiten

hacer esto. Además, los registros de eventos pueden ser extensos y los modelos de procesos generados pueden resultar complejos y difíciles de comprender.

Como una alternativa a estas limitaciones se ha estudiado la inclusión del procesamiento analítico en línea (*Online Analitical Processing*, OLAP) en la MP. Como resultado se ha dado a conocer el término de Cubos de Procesos (CP) que tiene como fin ayudar a comparar los datos de eventos y procesos, con el propósito de analizar sus variaciones, teniendo en cuenta diferentes dimensiones (por ejemplo: tiempo, lugar, entre otros). En investigaciones previas se abordan la noción de los CP, una de estas investigaciones sirvió de tesis de maestría a Tatinana Mamaliga (Mamaliga, 2013) de la universidad de Eindhoven y como resultado se obtuvieron celdas con información según sus dimensiones a partir de registros de eventos. Los CP permiten dividir registros de eventos muy grandes para realizar el análisis por partes y además permite considerar varios puntos de vista, como por ejemplo el monto total de las multas de una región determinada en un mes del 2015. Sin embargo, no es posible comparar varias celdas a nivel de datos en búsqueda de comportamientos similares o diferentes entre ellas. Esto limita el diagnóstico de los procesos, es decir, el análisis de desviaciones, procesos incompletos, patrones frecuentes, similitudes o diferencias entre subprocesos, temas que influyen en la calidad de las decisiones y en el tiempo de analizar los procesos de las organizaciones. Por ello se ha identificado como **problema a resolver**:

¿Cómo obtener información de los datos asociados a las celdas de procesos de manera que contribuya a disminuir el tiempo del análisis de los procesos y apoyar la toma de decisiones?

El trabajo de diploma está enmarcado dentro del **Objeto de estudio,** Minería de Procesos.

Para darle solución al problema planteado se ha propuesto como **Objetivo General**:

Implementar una extensión para la herramienta Cubo de Procesos que contribuya a disminuir el tiempo de análisis de los procesos y obtener evidencias que apoyen a la toma de decisiones.

De ahí que el trabajo de diploma se delimite al **Campo de acción,** Diagnóstico de Procesos.

Para alcanzar el objetivo general se proponen los siguientes **Objetivos Específicos:**

1. Realizar una fundamentación teórica donde se plasmen los principales enfoques de Minería de Procesos y cómo estos influyen en el Diagnóstico de Procesos. Además estudiar el estado del arte de los Cubos de Procesos.
2. Implementar una extensión para el Complemento ProCube que permita comparar celdas de un Cubo de Procesos de tal forma que disminuya el tiempo de análisis y mejore la toma de decisiones.
3. Realizar pruebas que validen que la extensión desarrollada funciona correctamente y que disminuye el tiempo y mejora el análisis.

Para darle cumplimiento a los objetivos se establecieron las siguientes **tareas**:

- Estudio de la MP, sus tipos, perspectivas y técnicas de diagnóstico para realizar un estado del arte que permita encaminar el proceso de investigación.
- Estudio y familiarización con la herramienta ProM para realizar análisis de MP y entender la estructura de sus complementos.
- Estudio de los CP y notaciones para modelar procesos con el objetivo de facilitar la implementación de las funcionalidades de la solución.
- Estudio y familiarización con las herramientas que trabajan con los CP para optimizar la implementación de la solución.
- Implementación de un sistema que permita comparar dos celdas y detectar deficiencias y sus causas en las ejecuciones de un proceso.
- Validación de la funcionalidad del sistema para comprobar que dicho sistema resuelve el problema planteado y satisface los requisitos deseados por el cliente.

Métodos teóricos empleados:

- Histórico-Lógico: este método es utilizado para realizar el estudio del estado del arte, analizar la evolución histórica de las técnicas de Minería de Procesos y las tendencias más recientes en el Diagnóstico de Procesos. Además la presente investigación tiene sus bases en investigaciones anteriores donde se han establecido los fundamentos que conforman la noción Cubo de Procesos y parte de la implementación ProCube donde se han plasmado las principales leyes para el funcionamiento de dicha noción.
- Modelación: la utilización de este método en la investigación se hace necesario toda vez que se requiere la representación abstracta de fenómenos complejos, como pueden ser procesos reales generados en organizaciones e incluso la propia noción de Cubo de Procesos que puede resultar difícil de comprender y manipular.
- Analítico-Sintético: se emplea para analizar las investigaciones previas al presente trabajo de diploma, así como para estudiar la documentación significativa para llevar a cabo la investigación y a partir de esta documentación se recopilan y sintetizan los elementos más importantes relacionados con la Minería de Procesos, el Diagnóstico de Procesos y los Cubos de Procesos.

Métodos Empíricos:

- Método de medición: este método se ve reflejado al aplicar las diferentes métricas sobre los requisitos funcionales, así como para medir los valores del tiempo y cantidad usados en diferentes gráficos comparativos.
- La entrevista: en la presente investigación se usa para capturar los diferentes requisitos de software deseados por el cliente.

Idea a defender: si se desarrolla una extensión para la herramienta Cubo de Procesos capaz de comparar celdas a nivel de datos es posible disminuir el tiempo de análisis y mejorar la calidad de dicho análisis.

Descripción de los capítulos

Capítulo 1. Fundamentación teórica:

En este capítulo se realiza un estudio de los principales conceptos relacionados con la Minería de Procesos y los Cubos de Procesos, así como el estado del arte de los mismos. Además se abordan los marcos de trabajo, lenguajes de programación y herramientas a utilizar.

Capítulo 2. Propuesta de Solución:

En este capítulo se describe la propuesta de solución a través de la presentación de diversos artefactos y la muestra del sistema resultante.

Capítulo 3. Validación de la solución:

En este capítulo se analizan los resultados y se aplican técnicas para evaluar si el sistema resultante cumple con el objetivo trazado al inicio de la investigación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

En el presente capítulo se hace un estudio del estado del arte de los principales conceptos relacionados con la Minería de Procesos y los Cubos de Procesos. El objetivo de este estudio es desvelar las ventajas que reporta la Minería de Procesos para la gestión de los procesos en las entidades y cómo los Cubos de Procesos contribuyen al análisis comparativo entre diferentes procesos. Conjuntamente se efectúa un estudio de las principales herramientas utilizadas, así como el lenguaje de programación Java.

1.2. Gestión de Procesos de Negocio (BPM)

La Gestión de los Procesos de Negocio incluye conceptos, métodos y técnicas para soportar el diseño, la administración, la configuración, la ejecución y análisis de los procesos de negocio. Dicha área se basa en la representación explícita de los procesos de negocio con sus actividades, encadenamiento y restricciones (Silverio Castro, 2013).

En cuanto a las principales funcionalidades que BPM provee se tienen (Club-bpm, 2014):

- Asignar actividades a las personas de forma automática y según cargas de trabajo.
- Recordar a las personas sus actividades, las cuales son parte de un flujo de trabajo.
- Optimizar la colaboración entre personas que comparten actividades.
- Automatizar y controlar el flujo de documentos, datos e imágenes.
- Asignarle proactivamente a las personas que deben ejecutar las actividades, todos los recursos necesarios (documentos, información, aplicaciones, etc.) en cada una de ellas.
- Definir y controlar alertas según criterios de tiempo, de evento o de condición, provocando así algún mensaje a un supervisor, un escalado de actividades a otras personas para que las resuelvan, y/o una resignación automática.
- Modificar cualquier instancia de proceso ya iniciada, sin necesidad de volver a iniciarla.
- Proveer una vista en línea para supervisores del estado e histórico de cada instancia de proceso, de cada actividad, y del desempeño de las personas.
- Hacer llegar a cada persona sus actividades y alertas, independientemente de su ubicación geográfica, a través de la WEB, Email, SMS, o cualquier otro dispositivo móvil.
- Proveer métricas para responsables de áreas, organizadores, gestores de procesos y calidad, tanto para mejora continua como para indicadores de calidad y de gestión.
- Se puede Integrar fácilmente con otros sistemas y aplicaciones.
- Proveer un alto nivel de soporte para la interacción humana.

La implantación del BPM, está emergiendo como un factor clave y estratégico, el cual las organizaciones están adoptando con más frecuencia para mejorar sus procesos y recursos empresariales. Constituyen uno de los principales ejes de inversión en las Empresas y

Administración Pública en los próximos años. La tecnología utilizada por BPM debe tener características específicas para ofrecer flexibilidad y agilidad en la evolución y dinamismo de los procesos de negocio y sistemas informáticos asociados. El primer requisito es que el proceso automatizado debe ser fácil de modificar sin ayuda de un programador, de forma que permita hacer frente a los cambios. Dicha tecnología ha evolucionado en esta dirección con la introducción de descripciones gráficas de los procesos, motores de reglas de negocio y otros mecanismos, y la posibilidad de modificar el proceso de forma inmediata, sobre la marcha y sin interrupciones. Para tener éxito en la implantación de este enfoque, las organizaciones no deben de cometer el gran error de centrarse solo en las tecnologías, sino en el conocimiento, dominio y mejora continua de sus procesos, datos y recursos empresariales (Fundación universitaria Konrad lorenz, 2014).

1.3. Minería de Procesos

Una disciplina estrechamente relacionada al BPM es la Minería de Procesos (MP), ya que se puede ver como el “eslabón perdido” entre la Minería de Datos y el BPM tradicional basado en modelos. La mayoría de las técnicas de Minería de Datos no están en absoluto centradas en procesos. Los modelos de procesos que potencialmente exhiben concurrencia son incomparables a las estructuras de Minería de Datos simples, tales como los árboles de decisión y las reglas de asociación. Por lo tanto, se necesitan nuevos tipos de representación y de algoritmos como los brindados por la MP (Manifiesto de Minería de Procesos, 2013).

El punto de partida de la MP es un registro de eventos (o trazas), de ahí que todas sus técnicas asuman que es posible registrar eventos secuencialmente tal que cada evento se refiera a una actividad (por ejemplo: un paso bien definido en algún proceso) y se relacione a un caso particular (una instancia de proceso). Los registros de eventos pueden almacenar información adicional acerca de los eventos (Manifiesto de Minería de Procesos, 2013). El crecimiento de un universo digital que está bien alineado con los procesos en las organizaciones hace posible registrar y analizar eventos. Los eventos podrían variar desde el retiro de dinero en efectivo, un doctor ajustando una máquina de rayos-X, un ciudadano solicitando una licencia de conducir, el envío de una declaración de impuestos hasta la recepción de un número de boleto electrónico por un viajero. El desafío está en aprovechar los datos de eventos en una forma significativa, por ejemplo, para proveer un mejor entendimiento, identificar cuellos de botella, anticipar problemas, registrar violaciones de políticas, recomendar contramedidas y simplificar procesos. La Minería de Procesos (MP) se propone hacer exactamente eso (Bose, 2012).

La MP es una disciplina de investigación relativamente joven que tiene como idea descubrir, monitorear y mejorar los procesos reales a través de la extracción de conocimiento de los registros de eventos ampliamente disponibles en los actuales sistemas de información. Además incluye el descubrimiento de procesos (extraer modelos de procesos a partir de un registro de eventos), la verificación de conformidad (monitorear desviaciones al comparar el modelo y el registro de eventos), la minería de redes sociales organizacionales, la construcción automática de modelos de

simulación, la extensión de modelos, la reparación de modelos, la predicción de casos y las recomendaciones basadas en historia (Buijs, y otros, 2011).

Las cuestiones típicas de la MP que pueden plantearse son (van der Aalst, 2011):

- ¿Cómo son realmente mis procesos?
- ¿Dónde están los cuellos de botella?
- ¿Se producen desviaciones de los procesos prescritos o descritos?

A fin de optimizar un proceso, en primer lugar se debe comprender la realidad del mismo, el proceso tal cual es. Y esto, normalmente, no es una tarea sencilla porque en los procesos de negocio están implicadas numerosas personas y a menudo se distribuyen entre diferentes unidades organizativas o incluso compañías.

1.3.1. Tipos de Minería de Procesos

Existen tres tipos de MP. El primero de ellos es el **descubrimiento**. Una técnica de descubrimiento toma un registro de eventos y produce un modelo sin usar ninguna información a-priori. Para muchas organizaciones es sorprendente ver que las técnicas existentes son realmente capaces de descubrir los procesos reales meramente basado en las muestras de ejecución en los registros de eventos (Manifiesto de Minería de Procesos, 2013). El descubrimiento es el área más atendida dentro de la MP, para la cual se han diseñado varios algoritmos. Se entiende por algoritmo de descubrimiento una función que mapea un registro de eventos hacia un modelo de proceso. Los modelos descubiertos pueden ser representados utilizando múltiples notaciones, dentro de las que se encuentran Redes de Petri y Redes de Flujo de Trabajo. El objetivo del descubrimiento de procesos es descubrir un modelo basado en un registro de eventos. Estos pueden tener toda clase de atributos (marcas de tiempo, información transaccional, uso de recursos, entre otros). Todos se pueden usar en el descubrimiento de procesos. Sin embargo, por simplicidad, se suele representar los eventos solo con nombres de actividades. De esta manera, un caso (una instancia de proceso) se puede representar por una traza que describe una secuencia de acciones (van der Aalst, 2011).

El segundo tipo es la **conformidad**. La MP no se limita al descubrimiento de procesos, de hecho, el proceso descubierto es tan sólo el punto de partida de un análisis más profundo. La comprobación de conformidad y la mejora relacionan modelo y registro. El modelo puede haber sido creado a mano o hallado por medio de descubrimiento de procesos. Para la comprobación de la conformidad, el comportamiento modelado y el comportamiento observado (el registro de eventos), se comparan. Existen varios enfoques para diagnosticar y cuantificar la conformidad. El enfoque más usado consiste en encontrar una alineación óptima entre cada traza en el registro y el comportamiento más parecido en el modelo (van der Aalst, 2011).

La verificación de conformidad puede ser usada para chequear si la realidad, tal como está almacenada en el registro de eventos, es equivalente al modelo y viceversa. Distintos tipos de modelos pueden ser considerados: la verificación de conformidad puede ser aplicada a modelos

procedurales, modelos organizacionales, modelos de procesos declarativos, políticas (reglas de negocio), regulaciones, entre otros. Las técnicas de verificación se emplean para detectar posibles desviaciones en las ejecuciones de los procesos.

El tercer tipo es el **mejoramiento**. Consiste en extender o mejorar un modelo de proceso existente usando la información acerca del proceso real almacenada en algún registro de eventos. Mientras la verificación de conformidad mide el alineamiento entre el modelo y la realidad, la mejora busca cambiar o extender el modelo a-priori. Por ejemplo, al usar marcas de tiempo en el registro de eventos, se puede extender el modelo para mostrar cuellos de botella, niveles de servicio, tiempos de procesamiento y frecuencias.

Un modelo de proceso no adecuado se puede corregir utilizando los diagnósticos proporcionados por la alineación. Si esta contiene muchos pasos, entonces podría tener sentido el salto de la actividad en el modelo. Además, los registros de eventos pueden contener información sobre recursos, marcas de fecha y datos del caso. Es posible analizar tiempos de espera entre actividades midiendo simplemente la diferencia de tiempo entre eventos causalmente relacionados y calculando estadísticas básicas tales como medias, varianzas e intervalos de confianza. Así es posible identificar los principales cuellos de botella (van der Aalst, 2011).

La siguiente imagen muestra los tres tipos de MP:

Figura 1 Tipos de Minería de Procesos: elaboración propia

1.4. Diagnóstico de Procesos

El ciclo de vida de la Gestión de Procesos de Negocio abarca las siete fases de un proceso de negocio y los sistemas de información asociados al mismo. Primeramente se diseña un proceso, el cual es convertido en un sistema ejecutable en la fase de implementación. En la fase de (re)diseño se crea un nuevo modelo de proceso o se adapta un modelo de proceso existente. En la fase de análisis se analiza un modelo candidato y sus alternativas. Después de la fase de (re)diseño, se implementa el modelo (fase de implementación) o se (re)configura un sistema existente (fase de

(re)configuración). En la fase de ejecución se ejecuta el modelo diseñado. Durante la fase de ejecución el proceso es monitoreado. Además, se pueden realizar pequeños ajustes sin rediseñar el proceso (fase de ajuste) (Daniel, y otros, 2013). En la fase de diagnóstico se analiza el proceso ejecutado y la salida de esta fase podría iniciar una nueva fase de rediseño del proceso como muestra la siguiente figura[1]:

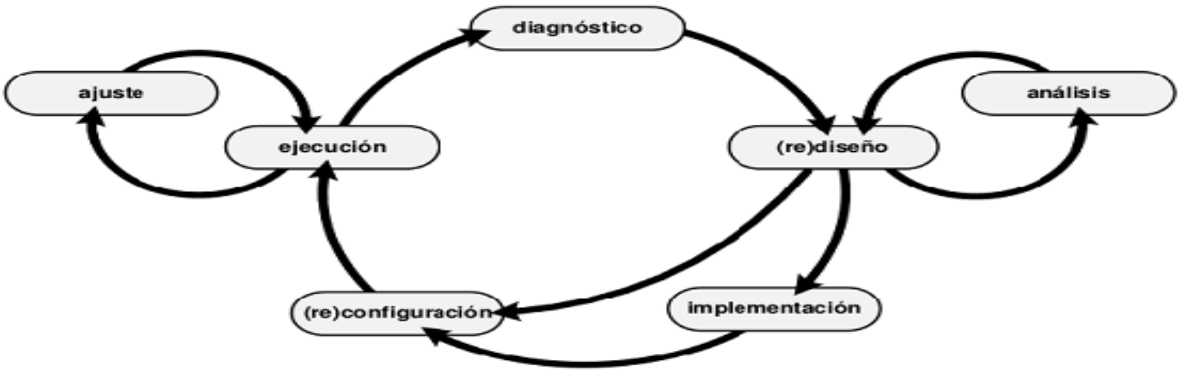

Figura 2 Etapas de la Gestión de Procesos de Negocio

Durante la fase de diagnóstico resulta importante determinar los patrones de control de flujo del proceso. Un proceso se puede descomponer en múltiples subprocesos utilizando los siguientes patrones de control de flujo (Pérez Alfonso, 2014):

- Secuencia: dos subprocesos se ejecutan secuencialmente si uno ocurre inmediatamente después del otro.
- Selección exclusiva: dos subprocesos forman parte de una selección exclusiva si, en un punto de decisión, se puede ejecutar solamente uno de los dos.
- Selección no exclusiva: dos subprocesos son opciones de una selección no exclusiva si, en un punto de decisión, pueden ejecutarse ambos o solamente uno de ellos.
- Paralelismo: dos subprocesos se ejecutan en paralelo si ambos se ejecutan simultáneamente.
- Lazo: dos subprocesos se encuentran en un lazo si se pueden repetir múltiples veces.

En este sentido, uno de los principales objetivos del diagnóstico es la determinación de patrones como el paralelismo y la selección, relacionados con la ejecución simultánea de actividades, debido al incremento de complejidad que significa su posterior análisis. Las técnicas para el Diagnóstico de Procesos se enfocan en identificar los comportamientos de mayor frecuencia de ocurrencia en el registro de eventos, la detección de ejecuciones anómalas y desviaciones en el proceso (Bose, 2012). La presencia de desviaciones en el modelo constituye un indicador de posibles fraudes o violaciones en las políticas establecidas, lo cual puede ser útil para las organizaciones a la hora de realizar auditorías. Las ejecuciones anómalas brindan información acerca de las posibles

[1] Figura obtenida de (Daniel, y otros, 2013)

violaciones del modelo. La identificación de los comportamientos y patrones frecuentes posibilita dirigir las técnicas de mejora hacia los elementos más críticos del proceso (Pérez Alfonso, 2014).

1.4.1. Técnicas para el Diagnóstico de Procesos

El Diagnóstico de Procesos y la MP se relacionan a través de diferentes técnicas que se describen a continuación. Las técnicas para realizar el Diagnóstico de Procesos se están incrementando considerablemente usando enfoques propios de la MP, de ahí que se materialicen como complementos del marco de trabajo ProM, que es la principal herramienta usada en esta disciplina de investigación.

1.4.1.1. Análisis de diagramas de puntos

El análisis de diagramas de puntos permite tener una visión general del registro de eventos, lo que facilita la identificación de los aspectos significativos en el proceso. Dicha técnica, similar a los diagramas de Gantt, presenta una “vista helicóptero” del registro de eventos. Muestra los eventos del proceso de forma gráfica, como puntos en un plano, donde una dimensión representa las trazas y la otra se refiere al instante en el que se ejecutaron los eventos (Song, y otros, 2009). El Análisis de diagramas de puntos presenta algunos indicadores de desempeño como el intervalo de tiempo mínimo, máximo y promedio entre la ejecución de los eventos, lo que facilita el análisis del rendimiento del proceso. Debido a que los eventos no se encuentran alineados con otras ejecuciones del proceso, se dificulta la detección de patrones que involucren varias trazas. Por otra parte, esta técnica es susceptible a la cantidad de actividades presentes en el proceso, pues al incrementarse el número de actividades al orden de las decenas, la inspección manual, la comprensión de la tabla de puntos y la identificación de patrones se torna compleja. Se considera que todos los eventos pertenecen al mismo nivel de abstracción en el registro de eventos, lo que impide la identificación de los subprocesos (Pérez Alfonso, 2014).

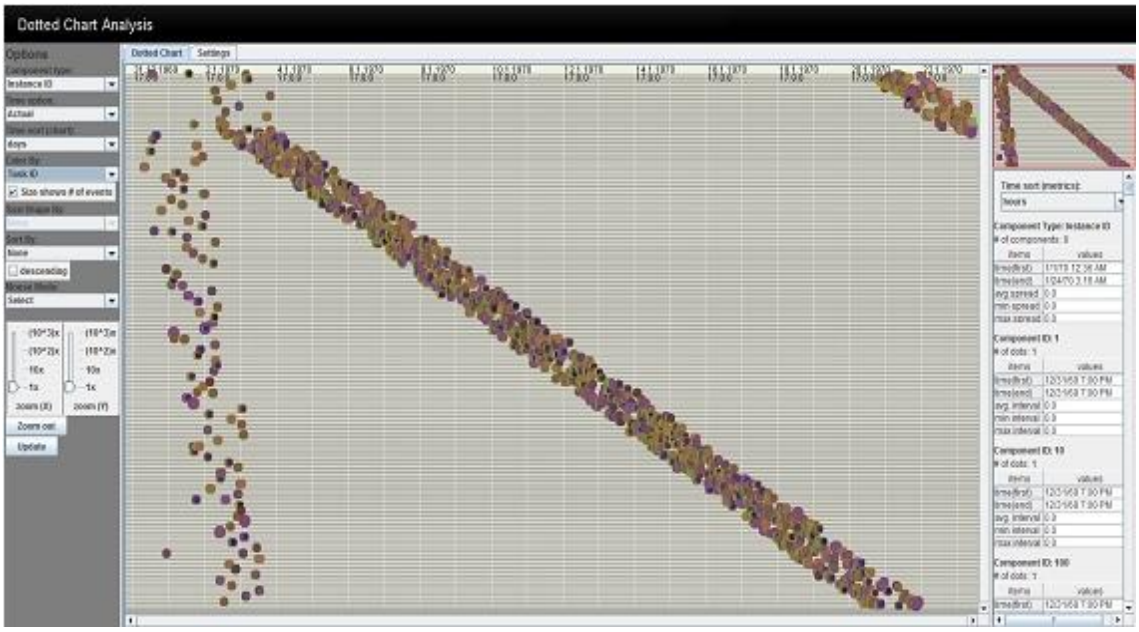

Figura 3 Diagramas de puntos

Cada uno de los eventos se representa en el diagrama con un color particular como se muestra en la figura[2] anterior generada a partir del marco de trabajo ProM.

1.4.1.2. Arreglos Tándem

Con frecuencia se usan los arreglos Tándem y repeticiones máximas para capturar patrones recurrentes dentro y a través de las trazas. Esta técnica se emplea para procesar el registro de eventos, sustituyendo determinados eventos por los respectivos patrones a los que pertenecen. El registro de eventos una vez procesado es utilizado para aplicar algoritmos de descubrimiento de procesos (Jagadeesh Chandra Bose, y otros, 2009).

Esta técnica permite la determinación de patrones en tiempo de cómputo lineal, aunque el número de patrones descubiertos puede ser elevado. Por otra parte, los patrones descubiertos son atómicos por lo que las dependencias entre estos deben ser descubiertas por separado.

1.4.1.3. Alineación de Trazas

El objetivo de esta técnica es alinear trazas de manera que los registros de eventos puedan ser explorados fácilmente. Puede ser usada para explorar los procesos en etapas tempranas de análisis y para responder preguntas específicas que surjan en etapas posteriores del análisis. De ahí que se complemente con las técnicas existentes de MP centrándose en el descubrimiento y el chequeo de conformidad (Jagadeesh Chandra Bose, y otros, 2011).

Esta técnica está inspirada en la alineación de secuencia biológica que es una herramienta fundamental en la bioinformática que ayuda a determinar las estructuras secundarias y terciarias de proteínas y moléculas, su evolución y funciones (Waterman, 2000). Similar a la bioinformática, la MP se encuentra relacionada con las secuencias, aunque en este caso las secuencias se originan por el registro de la ejecución del proceso. Para realizar la Alineación de Trazas se emplea el enfoque de alineación progresiva. El mismo se basa en la construcción iterativa de parejas de alineaciones, permitiendo la alineación entre: dos alineaciones, una traza y una alineación o dos trazas (Jagadeesh Chandra Bose, y otros, 2011). La calidad de la alineación final depende en gran medida del orden en que se realiza la alineación progresiva.

Una de las potencialidades de la Alineación de Trazas consiste en posibilitar la identificación de desviaciones en la ejecución del proceso. Dicha técnica permite la visualización de múltiples ejecuciones del proceso simultáneamente y facilita la determinación visual de los patrones de control de flujo (Pérez Alfonso, 2014).

Sin embargo, esta técnica no permite detectar los subprocesos que conforman al proceso analizado y enmarcar en estos las anomalías y patrones identificados. Esto dificulta la contextualización del aspecto detectado y el entendimiento de las causas que lo originaron. Por otra parte, la presencia

[2] Figura tomada de (Pérez Alfonso, 2014)

de ruido en el registro de eventos puede conducir a la obtención de alineaciones con baja calidad (Yzquierdo Herrera, y otros, 2013).

La siguiente figura[3] muestra un conjunto de trazas alineadas:

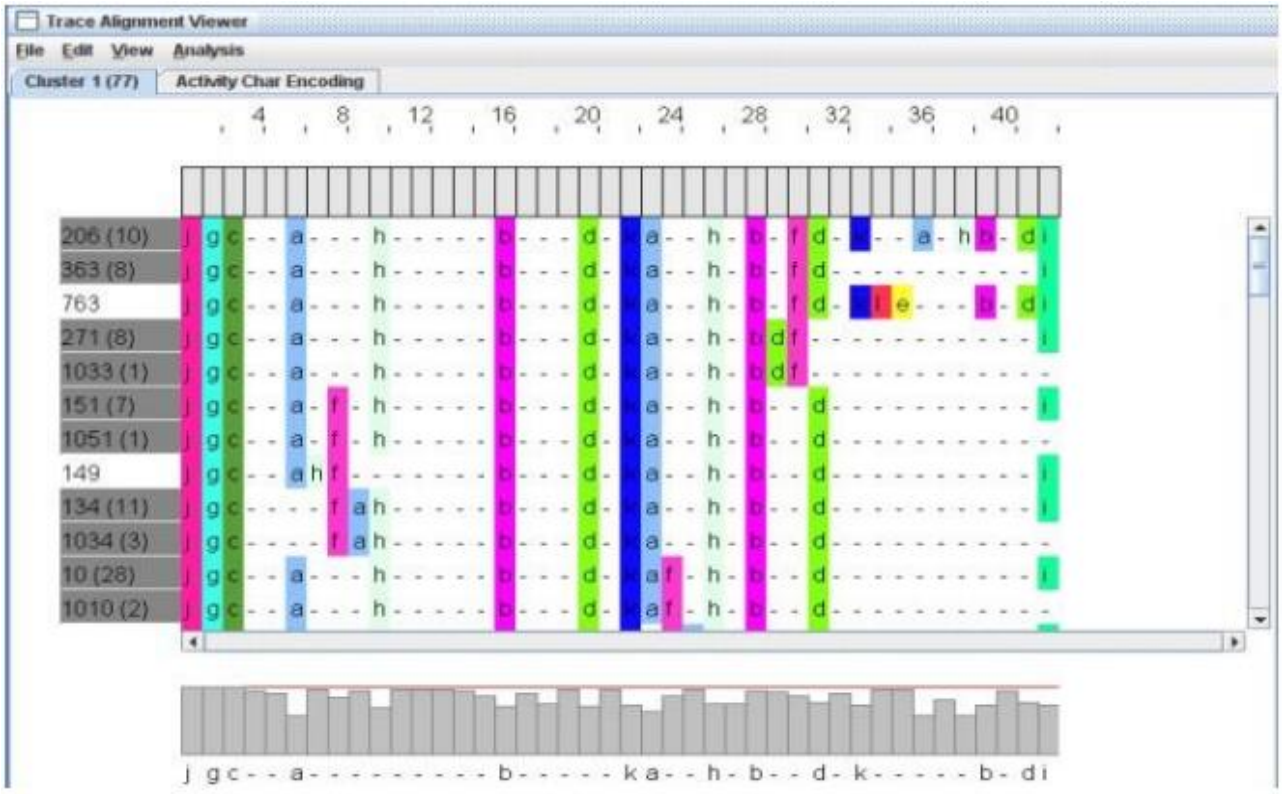

Figura 4 Alineación de Trazas

1.4.1.4. Minería Difusa

La Minería Difusa es una técnica de diagnóstico que muestra las actividades y sus relaciones según diferentes niveles de abstracción. Utiliza similitudes con la forma de representar la información en la cartografía, aplicando conceptos como: abstracción, personalización, agregación y énfasis (van der Aalst, 2011). Dicha técnica produce un modelo basado en grafos, donde existen dos tipos de nodos: los que representan una actividad y los que agrupan un conjunto de actividades denominados clústeres.

La agregación se refiere a limitar la cantidad de elementos, nodos y relaciones que se muestran. En su lugar, varios elementos forman clústeres encapsulando la información y facilitando su visualización. La cantidad de información que se muestra es controlada mediante el empleo de umbrales de permisibilidad. La abstracción consiste en omitir de la visualización la información que en determinado contexto es insignificante. En determinados componentes se destacan características como color, contraste, saturación y tamaño para facilitar la visualización de la

[3] Tomada de (Yzquierdo Herrera, y otros, 2013)

información relevante. Por su parte la personalización permite configurar la información que se muestra de acuerdo a determinados criterios (Pérez Alfonso, 2014).

A continuación se presenta un modelo obtenido a partir de la aplicación de esta técnica:

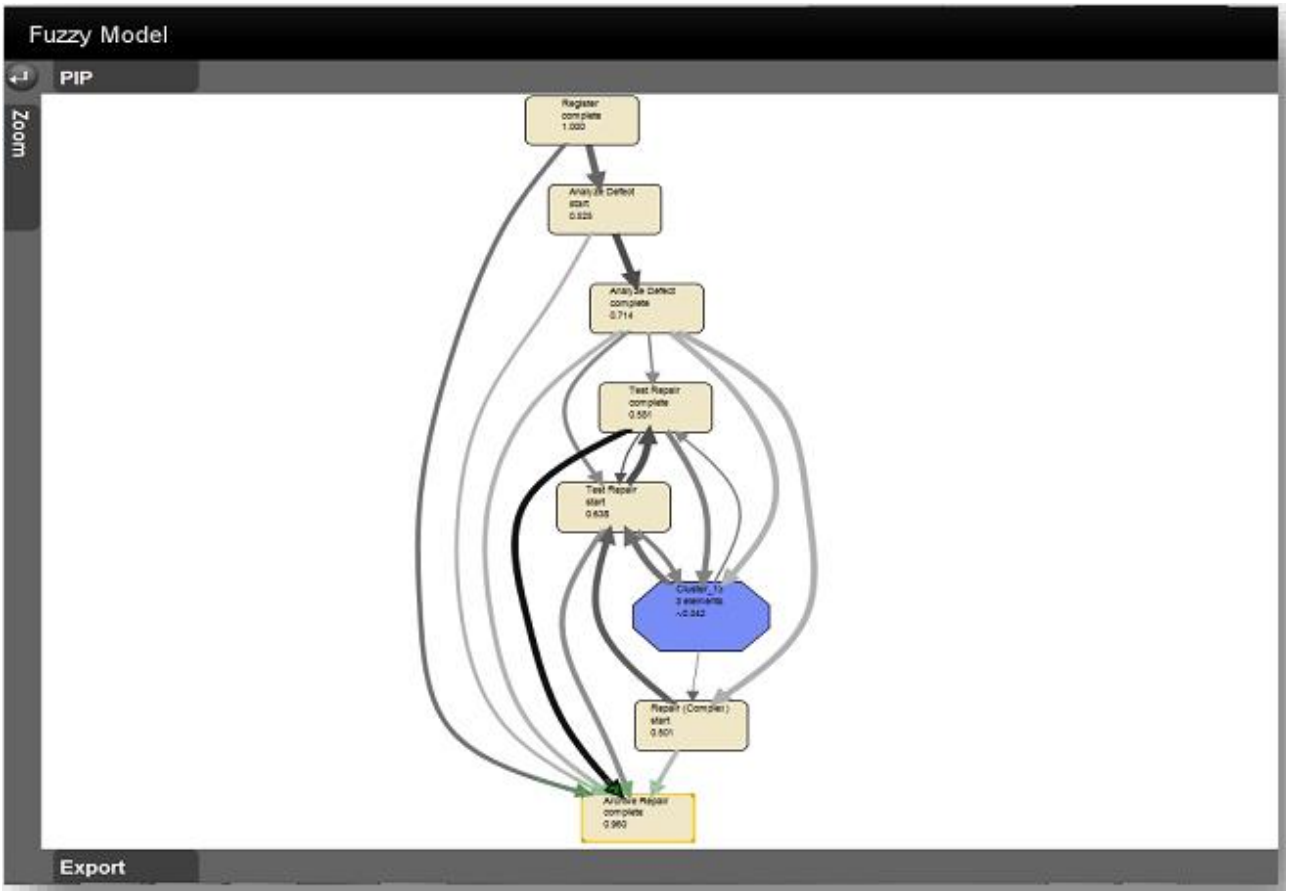

Figura 5 Modelo obtenido al aplicar Minería Difusa: elaboración propia.

1.5. Cubos OLAP

El procesamiento analítico en línea (*Online Analitical Processing*, OLAP) es un método para ayudar a la toma de decisiones en situaciones donde un conjunto de datos de difícil medición, tales como ventas o ganancias, necesiten ser analizados en diferentes niveles de agregación estadística. Introducido en 1993 por Codd como un nombre más genérico para el análisis de datos multidimensionales, OLAP adopta el paradigma multidimensional como un medio para proveer un acceso rápido a datos cuando el análisis se realiza desde diferentes perspectivas (Mamaliga, 2013).

En estos modelos los datos son vistos como cubos que consisten en categorías descriptivas (dimensiones) y valores cuantitativos (medidas). Los datos son clasificados en diferentes dimensiones y pueden ser vistas unas con otras en diferentes combinaciones para obtener diferentes análisis de los datos que contienen. El modelo multidimensional de datos simplifica a los usuarios acciones como: formular consultas complejas, arreglar datos en un reporte, cambiar datos resumidos a datos detallados, entre otros (Ibarra, 2005).

Gracias a la incorporación de las bases de datos de tipo multidimensional y al nacimiento del nuevo concepto Cubo OLAP, las herramientas de soluciones para sistemas de inteligencia de negocios han avanzado notablemente en cuanto a las prestaciones que estas aplicaciones brindan a las empresas, donde la información confiable, precisa y en el momento oportuno, son uno de los bienes más preciados (Informática-Hoy, 2014).

Las operaciones básicas de un Cubo OLAP son (Ibarra, 2005):

- *Roll-Up*: este comprende el conjunto de datos. Esto puede involucrar acumulaciones simples o agrupaciones complejas que incluyen datos interrelacionados.
- *Drill-Down*: OLAP puede moverse en la dirección contraria y presentar automáticamente datos detallados que abarcan datos consolidados
- *Slicing y Dicing*: se refiere a la capacidad de visualizar a la base de datos desde diferentes puntos de vista

La utilización de Cubos OLAP tiene dos ventajas fundamentales (Mamaliga, 2013):

- Facilidad de uso: una vez construido el cubo, el usuario de negocio puede consultarlo con facilidad, incluso si se trata de un usuario con escasos o nulos conocimientos técnicos. La estructura jerárquica es sumamente fácil de comprender para la mente humana, y si esta coincide con el modelo de negocio, los resultados suelen ser espectaculares, ya que el cubo se convierte en una gran "tabla dinámica" que el usuario puede consultar en cualquier momento.

- Rapidez de respuesta: habitualmente el cubo tiene pre-calculados las distintas agregaciones, por lo que los tiempos de respuesta son muy cortos. Si está bien diseñado, el cubo tendrá un tiempo de respuesta pequeño aunque la cantidad de datos sea grande.

En comparación con su contrapartida de procesamiento transaccional en línea (*Online Transactional Processing*, OLTP), OLAP está optimizado para el análisis de los datos, más bien que para almacenar datos originados desde múltiples fuentes para evitar la redundancia. Por tal motivo, OLAP está basado mayormente en datos históricos y no en datos instantáneos, los cuales son bastante complicados de analizar, ordenar, agrupar o comparar (van der Aalst, 2013).

1.6. Cubos de Procesos

La noción de Cubos de Procesos (CP) está relacionada con el enfoque de “divide y vencerás” en la MP donde grandes registros de eventos son particionados para formar registros más pequeños con el objetivo de mejorar el desempeño.

Un CP es una estructura multidimensional construida a partir de registros de eventos de manera que facilite un análisis de MP más exhaustivo. Además está compuesto por un conjunto de celdas de procesos (van der Aalst, 2012) y la principal diferencia con un Cubo OLAP radica en las

características de sus celdas. A diferencia de los Cubos OLAP, no existe una medida real del interés de cuantificar una operación de negocio. Mientras las estructuras OLAP están diseñadas para análisis de operaciones de negocio, los CP optan por el análisis de procesos. Por tanto, cada dimensión de análisis está compuesta por atributos de eventos.

El contenido de una celda en el CP cambia de números reales a eventos. Mientras en OLAP, las dimensiones de análisis son usadas para poblar el cubo, en los CP los eventos de un registro de eventos son usados para crear dimensiones de análisis (Mamaliga, 2013).

La figura 6[4] muestra un cubo de tres dimensiones. No obstante, un CP puede tener cualquier cantidad de dimensiones y estas dimensiones pueden estar basadas en cualquier propiedad de los eventos. En la figura se puede observar que los eventos están agrupados en celdas basadas en "tipo de caso" (*case type*), "clase de eventos" (*event class*) y "tiempo" (*time window*). Cada celda se refiere al conjunto de todos los eventos que pertenecen al tipo de "clase de eventos" (ec) y a un "tiempo" (tw).

La dimensión de "tipo de caso" está basada en propiedades de los casos en general y no en las características individuales de los eventos. De ahí que si un evento "e" es de "tipo de caso" entonces todos los eventos del caso al cual "e" pertenezca también pertenecen al "tipo de caso". El "tipo de caso" puede estar basado en un tipo específico de cliente como puede ser "dorado" o "plateado" o en la cantidad total (ejemplo: < 1000 o >1000).

La dimensión "clase de evento" está basada en las propiedades de los eventos individuales, por ejemplo el nombre de actividad, recurso asociado o localización geográfica. Esta dimensión puede depender del nombre de actividad, por ejemplo, podrían existir tres clases de eventos basados en la superposición de conjuntos de nombres de actividades: {A, B}, {C, D}, y {E}.

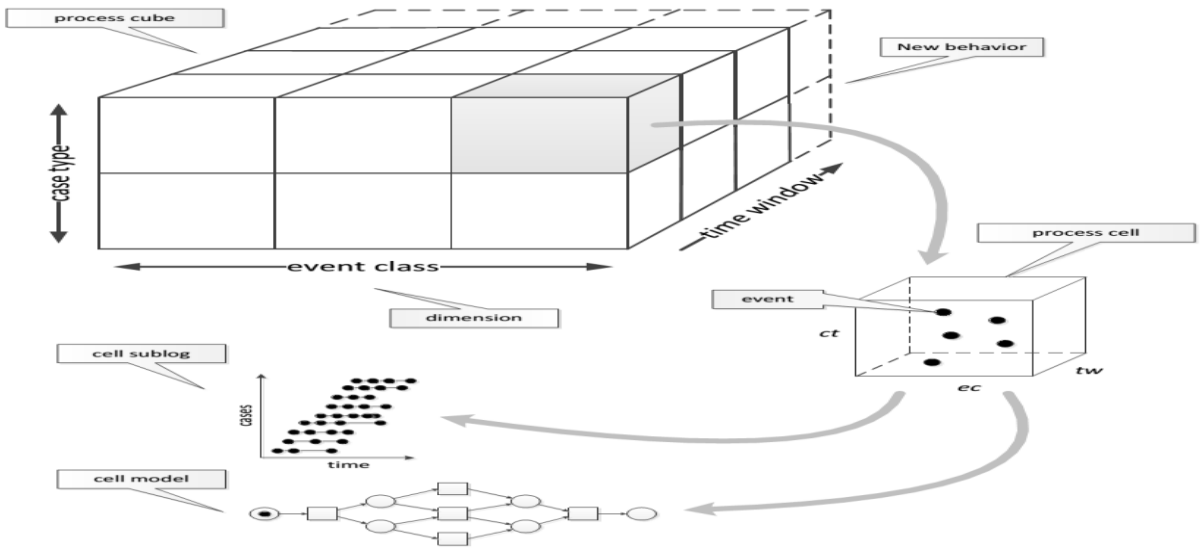

Figura 6 Componentes de un Cubo de Procesos

[4] Tomada de (van der Aalst, 2013)

La dimensión “tiempo” se puede referir a un período, por ejemplo, puede referirse a todos los eventos que tuvieron lugar en diciembre 2014 (van der Aalst, 2013).

Como se puede observar un CP está compuesto por un conjunto de celdas de procesos y cada celda puede tener un modelo de proceso predefinido o descubierto a partir del registro de evento exclusivo de la propia celda. De esta forma se pueden entender las diferencias entre los clientes “dorados” y los “plateados”, entre los meses Diciembre y Enero o simplemente entre dos personas a la hora de realizar una operación.

1.6.1. Operaciones en los Cubos de Procesos

Al igual que en los Cubos OLAP, en los Cubos de Procesos se pueden aplicar cuatro operaciones básicas que se diferencian entre sí, pero todas contribuyen a mejorar el análisis.

1.6.1.1. Slice y Dice

La operación “*Slice*” produce un cubo “rebanado” permitiendo al analista seleccionar un valor específico para una de las dimensiones. Por ejemplo, para analizar las multas se puede rebanar el cubo por una determinada ubicación geográfica (ejemplo: Milán). Al hacer esto la dimensión correspondiente a la ubicación geográfica es removida del cubo y sólo se consideran las multas que tienen lugar en Milán. Al hacer lo mismo para el año 2014, se elimina la dimensión “tiempo” y solo se consideran las multas de ese año.

La operación “*Dice*” produce un sub-cubo después de que el analista seleccione valores específicos para múltiples dimensiones. Si se aplica la operación “*Dice*” para los años 2013 y 2014 y las ubicaciones geográficas de Milán y Roma, no se remueve ninguna dimensión, pero sólo se consideran las multas llevadas a cabo en Milán y Roma (van der Aalst, 2013).

1.6.1.2. Roll-Up y Drill-Down

Las operaciones “*Roll-Up*” y “*Drill-Down*” no remueven ningún evento pero cambian el nivel de granularidad de una dimensión particular. Por ejemplo, antes de aplicar “*Drill-Down*” a la dimensión tiempo esta estaría formada así: Dimensión[5] (tiempo) = {T2011; T2012; T2013} y después de aplicar dicha operación quedaría: Dimensión (tiempo) = {T2011; T-Ene-2012; T-Feb-2012;…; T-Dec- 2012; T2013}. Esta operación equivale a dividir (*split*) las dimensiones.

La operación “*Roll-Up*” es todo lo contrario, equivale a agrupar (*merge*). Por ejemplo: Dimensión (tipo de cliente) = {{dorado}; {plateado}} es agrupado de la siguiente forma: Dimensión (tipo de cliente) = {{dorado; plateado}} (van der Aalst, 2013).

[5] Dimensión (valor) se refiere a un conjunto de valores de tipo “valor” que constituyen una dimensión del CP

1.7. Metodología para comparar celdas de Cubos de Procesos

A pesar de los múltiples esfuerzos realizados para integrar la MP con los Cubos OLAP todavía es un reto lograr una comparación que aporte información significativa para los negocios. En este sentido existe una metodología para comparar diferentes celdas de los CP mediante la aplicación de las operaciones dadas a conocer en el epígrafe anterior (Gupta, y otros, 2015).

Como se menciona en el epígrafe anterior los CP son una estructura de datos multidimensional donde los eventos y modelos de procesos están organizados usando diferentes dimensiones para facilitar la comparación entre múltiples procesos (van der Aalst, 2013) (Mamaliga, 2013). Para conformar un CP es necesario utilizar tres elementos: la Visualización de los Cubos de Procesos (*Process Cube View*, PCV), la Estructura de los Cubos de Procesos (*Process Cube Structure*, PCS) y la Base de Eventos (*Event Base*, EB). PCV se define a través del PCS y el EB. PCS es el encargado de definir las dimensiones del cubo y el EB está compuesto por el conjunto de atributos que se utilizarán como dimensiones del mismo.

A través de la manipulación de estos elementos se puede aplicar la metodología antes mencionada, la cual se representa en la siguiente imagen y está compuesta por tres etapas: Extracción y transformación de los datos, Visualización de los CP y operaciones OLAP sobre los mismos, y Aplicación de la Minería de Procesos a la salida de la etapa anterior.

Figura 7 Metodología para comparar celdas de un CP: elaboración propia

En este caso se toman valores específicos a modo de ejemplo.

1. Extracción y transformación de los datos

El objetivo de esta etapa es extraer los datos de diferentes repositorios (en este caso ITS[6], PCR[7] y VCS[8]) utilizando herramientas como JSON-RPC y XML-RPC para integrarlos en un sistema único como una Base de Eventos que contiene dos tipos de atributos:

[6] ITS: Issue Tracking System (Sistema de Seguimiento a los Problemas)

[7] PCR: Pedagogical Code Review (Revisión Pedagógica de Código)

[8] VCS: Version Control System (Sistema de Control de Versiones)

- Atributos de Caso: son las propiedades asociadas al caso, las cuales se mantienen constantes para todos los eventos pertenecientes al mismo caso. Por ejemplo: CasoID, Tipo, Estado, Estatus, Prioridad, Sistema Operativo (SO), Reportero, Propietario, Componente y Tiempo Reportado para un problema.
- Atributos de Eventos: propiedades que describen a cada evento como Actividad, Actor (Recurso) y Marca de Tiempo (cuándo se realiza la actividad).

2. Visualización de los CP y operaciones OLAP

A diferencia de los Cubos OLAP, donde cada celda es una medida numérica, una celda de CP está compuesta por el conjunto de los eventos provenientes de una Base de Eventos basados en valores de dimensiones. A su vez, todas las propiedades almacenadas en una Base de Eventos pueden ser una dimensión para el cubo. Para la visualización se aplican técnicas como Agrupamiento, Jerarquías y Selección de Sub-Conjuntos (Gupta, y otros, 2015). Luego de lograr la visualización, se está en condiciones de aplicar las operaciones OLAP para comparar determinadas celdas, tal como muestra la figura siguiente, donde se muestran tres dimensiones (Estado, Estatus y Tipo de Problema) con sus respectivos atributos[9]:

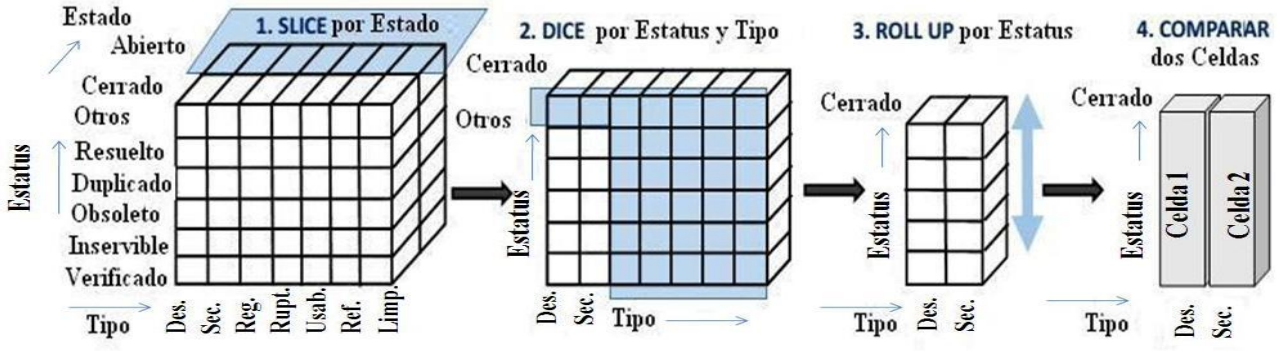

Figura 8 Aplicación de operaciones OLAP sobre un CP: elaboración propia

Después de aplicar una secuencia de operaciones se obtienen celdas que pueden ser comparadas y que contienen variantes de procesos, como se muestra en las celdas de la derecha de la figura anterior correspondientes a los Tipos de Problemas de Desempeño (Des.) y de Seguridad (Sec.).

3. Aplicación de la Minería de Procesos

A las celdas obtenidas en la etapa anterior se le aplican las diferentes técnicas de MP para mejorar la comparación. Por ejemplo, si se tiene en cuenta la frecuencia con que ocurrió cada Tipo de Problema resultante del análisis anterior se obtienen los modelos presentados en la figura 9. En esta figura el modelo de la izquierda corresponde a los problemas en el Desempeño y el modelo de la derecha a los problemas en la Seguridad. Además, el grosor de las líneas, conjuntamente con los números asociados a ellas, corresponde a la frecuencia con que se ejecutan esas secuencias de actividades. De igual manera, el sombreado de los nodos corresponde a la frecuencia absoluta

[9]Tipo: Desempeño, Seguridad, Regresión, Ruptura, Usabilidad, Refinamiento, Limpieza.

con que se ejecuta la actividad asociada a él. Otros análisis similares pueden hacerse sobre los modelos resultantes de dos celdas como, por ejemplo, encontrar cuellos de botellas[10] o redes sociales para encontrar la importancia de las personas vinculadas al proceso.

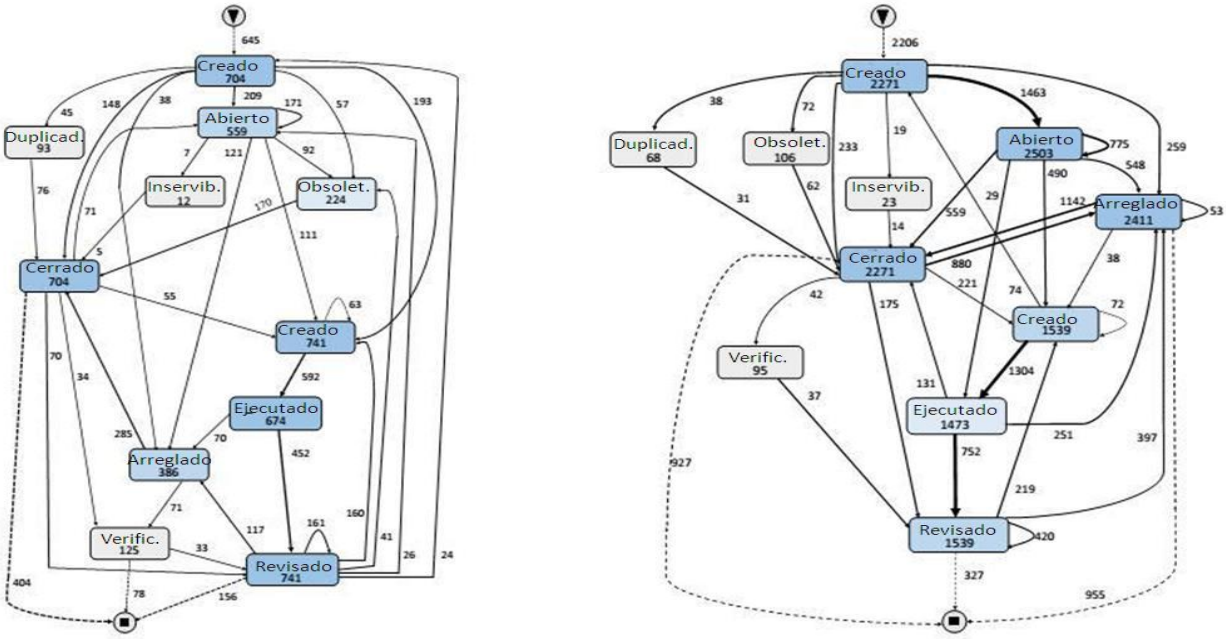

Figura 9 Modelos de dos celdas de procesos: elaboración propia

1.8. Lenguajes y Herramientas

1.8.1. Lenguaje Unificado de Modelado

El Lenguaje Unificado de Modelado (*Unified Modeling Language*, UML), permite la representación de sistemas de software en un lenguaje común, es uno de los más conocidos y utilizados en la actualidad. Se utiliza para definir un sistema en cada una de las etapas por las que tiene que transitar, indica lo que supuestamente hará, no cómo lo hará. Incluye aspectos conceptuales tales como: procesos de negocio, funciones del sistema y aspectos concretos como: expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables, por lo que es muy útil para comprender el diseño del sistema (Booch, y otros, 2006). Permite modelar diferentes tipos de aplicación que se ejecuten en cualquier tipo de combinación de hardware, sistema operativo, lenguaje de programación o infraestructura de red. Está diseñado para ser usado sobre los fundamentos de la orientación a objetos, convirtiéndolo en un marco ideal para los lenguajes orientados a objetos como C++, Java y C#, teniendo uso limitado para otros paradigmas de programación. Independientemente de la metodología que se utilice para llevar a cabo el análisis y el diseño, se puede utilizar UML para expresar los resultados, característica que lo ha llevado a ser ampliamente difundido (Watson, 2013).

[10]Lugar en el proceso donde ocurre un estancamiento o retrasos de las tareas. Identificarlo es vital para disminuir el tiempo de duración de un proceso.

1.8.2. Java

Java es un lenguaje de programación de alto nivel, orientado a objetos, el cual tiene la capacidad de ser ejecutado en una gran cantidad de dispositivos. En su mayor parte la sintaxis de Java está influenciada por el popular lenguaje C y por Visual Basic, aunque deja de lado algunas características de bajo nivel que tiene C como son el manejo de memoria y los apuntadores (punteros). Para las gestiones de manejo de memoria Java utiliza una aplicación llamada Recolector de Basura (*garbaje collector*) el cual se encarga de limpiar la memoria.

La utilización de Java está influenciada principalmente, además de las características antes mencionadas, que constituyen obvias ventajas, por las facilidades para el desarrollo que brinda. Además posee todas las características del paradigma de la programación orientada a objetos: herencia, encapsulamiento, abstracción, polimorfismo, modularidad entre otras que son de utilidad a la hora de representar contenidos abstractos. También cuenta con todas las estructuras de control y bifurcaciones que poseen la mayoría de lenguajes (JavaWorld, 2012).

1.8.3. MySQL

MySQL es el Sistema de Administración de Bases de datos (*Database Management System*, DBMS) más popular, desarrollado y proporcionado por la empresa sueca MySQLAB. Es un Sistema de Gestión de Bases de datos relacional multihilo y multiusuario que fue escrito en C y C++ y destaca por su gran adaptación a diferentes entornos de desarrollo, permitiendo su interacción con los lenguajes de programación más utilizados como PHP, Perl y Java y su integración a distintos sistemas operativos. También es muy destacable, la condición de “código abierto” que tiene MySQL, lo que permite que su utilización sea gratuita e incluso se pueda modificar con total libertad, pudiendo descargar su código fuente. Esto ha favorecido positivamente en su desarrollo y continuas actualizaciones, para hacer de esta herramienta una de las más utilizadas por los programadores orientados a Internet (Perez García, 2007).

Algunas de las **ventajas** que posibilitan que MySQL se encuentre en la preferencia de los usuarios son (MySql.com, 2015):

- Velocidad al realizar las operaciones, esto lo convierte en uno de los gestores con mejor rendimiento.
- Bajo costo en requerimientos para la elaboración de bases de datos, ya que debido a su bajo consumo puede ser ejecutado en una máquina con escasos recursos sin ningún problema.
- Facilidad de configuración e instalación.
- Baja probabilidad de corromper datos, incluso si los errores no se producen en el propio gestor, sino en el sistema en el que está.
- Soporta gran cantidad de tipos de datos para las columnas.

- Cada base de datos cuenta con tres archivos: uno de estructura, uno de datos y uno de índice. Además soporta hasta 32 índices por tabla.
- Flexible sistema de contraseñas y gestión de usuarios, con un muy buen nivel de seguridad en los datos.
- El servidor soporta mensajes de error en distintas lenguas.
- Está desarrollado en C/C++.
- La API se encuentra disponible en C, C++, Eiffel, Java, Perl, PHP, Python.
- Es altamente confiable en cuanto a estabilidad se refiere.
- Facilidad de uso. Es un sistema de base de datos de alto rendimiento pero relativamente simple y es menos complejo de configurar y administrar que sistemas más grandes.

1.8.4. ProM

La herramienta ProM es un marco de trabajo extensible y de código abierto para el trabajo con algoritmos de MP. Soporta una amplia variedad de técnicas para dicha finalidad y además es multiplataforma e implementado en Java (B. F. van Dongen, 2005).

La siguiente figura muestra una idea general del marco de trabajo ProM[11].

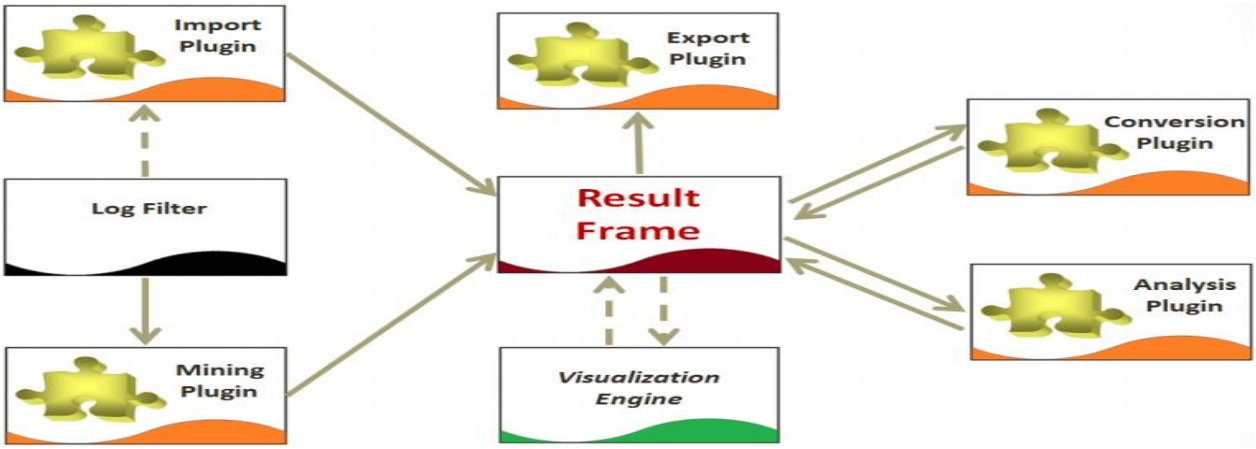

Figura 10 Estructura del marco de trabajo ProM

Aquí se incluyen los principales tipos de Complementos que conforman la aplicación y la relación entre ellos. Antes de aplicar cualquier técnica de MP, las trazas de eventos deben ser filtradas usando un filtro de trazas. Luego las trazas filtradas pueden ser minadas usando el Complemento para la minería y después deben ser almacenadas como un marco de resultado. El motor de visualización asegura que los resultados puedan ser visualizados correctamente. Una traza de eventos, así como diferentes modelos, pueden ser cargados al marco de trabajo mediante el Complemento de importación. El Complemento de conversión se especializa en convertir el resultado a un formato diferente y el de análisis procesa el resultado desde diferentes perspectivas, ambos utilizan resultados de minería como entrada.

[11]Según (B. F. van Dongen, 2005)

El marco de trabajo ProM que cuenta con funciones predefinidas dentro del marco para viabilizar el desarrollo de Complementos, es la herramienta que ofrece mayores facilidades para el trabajo en la MP, así como el entorno más completo en cuanto a funcionalidades que ofrece para implementar técnicas de MP. Razones por las cuales este y otros trabajos están desarrollados en este marco. Además pretende establecerse como la plataforma estándar para la MP mediante el establecimiento de una comunidad activa y reconocida de contribuidores y usuarios (TU/e, 2015).

1.8.5. Eclipse

Eclipse es un entorno de desarrollo integrado (*integrated development enviroment*, IDE), de código abierto y multiplataforma, así como una potente plataforma de programación, desarrollo y compilación de elementos tan variados como sitios web, programas en C++ o aplicaciones Java. Dispone de un editor de texto con un analizador sintáctico. La compilación es en tiempo real. Tiene pruebas unitarias con JUnit, control de versiones con CVS[12], integración con Ant[13], asistentes (*wizards*) para creación de proyectos, clases, etc. Además, a través de complementos (Complementos) libremente disponibles, es posible añadir control de versiones con Subversion e integración con Hibernate.

El entorno de desarrollo integrado de Eclipse emplea Complementos para proporcionar toda su funcionalidad al frente de la plataforma de "cliente enriquecido[14]", a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Además en el SDK de Eclipse se provee soporte para Java y CVS y no tiene por qué ser usado únicamente con estos lenguajes, ya que soporta otros lenguajes de programación. De la misma forma, el SDK de Eclipse incluye las herramientas de desarrollo de Java, ofreciendo un IDE con un compilador de Java interno y un modelo completo de los archivos fuente de Java. Esto permite técnicas avanzadas de refactorización y análisis de código. Mediante diversos Complementos estas herramientas están también disponibles para otros lenguajes como C/C++ (Eclipse CDT) y en la medida de lo posible para lenguajes de script no tipados[15] como PHP o Javascript. El IDE también hace uso de un espacio de trabajo, en este caso un grupo de metadatos en un espacio para archivos plano, permitiendo modificaciones externas a los archivos en tanto se refresque el espacio de trabajo correspondiente (Eclipse.org, 2015).

[12] CVS: Concurrent Versions System

[13] ANT: Another Neat Tool

[14] Plataforma que utiliza solo las funcionalidades necesarias para realizar determinado objetivo.

[15] Lenguajes donde no es necesario definir los tipos de datos de las variables.

1.8.6. Palo

Palo es un motor de Procesamiento Analítico Multidimensional en Línea (MOLAP por sus siglas en inglés), creado por la empresa alemana Jedox, que permite crear y almacenar datos, así como generar informes en hojas de cálculo, vía Microsoft Excel, Open Office Calc o en el navegador web. La plataforma de Palo está basa en componentes de código abierto y trabaja fundamentalmente con grandes volúmenes de datos contenidos en hojas de cálculo, permitiendo a varios usuarios compartir un almacenamiento centralizado de dichos datos. Los datos en la base de datos de Palo se almacenan en forma cubo en el Palo OLAP Server. Palo soporta hasta 256 dimensiones con jerarquías dentro de cada dimensión (Taniar, y otros, 2011).

Palo Suite está compuesto por las siguientes aplicaciones básicas:

- Palo OLAP Server: servidor OLAP multidimensional con tecnología "en memoria", que organiza los datos en forma de Cubos, dimensiones, elementos y atributos de elementos.
- Palo para Excel: extensión de MS Office Excel, que permite el acceso a las funcionalidades de Palo OLAP Server y a su poderosa base de datos multidimensional, directamente desde Excel.
- Palo ETL Server: herramienta de adquisición de datos basado en la Web que extrae, transforma y carga datos de los sistemas transaccionales, almacenes de datos y otras fuentes externas.
- Palo Web: es una interfaz Web que permite acceder a los datos de los cubos sin tener software instalado. A través de la misma los usuarios pueden publicar informes y cuadros de control predefinidos así como crear sus propios análisis seleccionando las informaciones que van a publicarse, los ejes de análisis, entre otros.

Entre algunas de las ventajas de utilizar el motor Palo se incluyen

- Tiene una estructura centralizada con una base de datos multidimensional.
- Puede llegar a ser hasta 100 veces más rápido que un sistema ROLAP.
- Permite la modelación en tiempo real desde Excel o desde el navegador Web.

La etapa de procesamiento y carga de datos puede ser bastante larga, sobre todo para grandes volúmenes de datos debido a que Palo es un sistema MOLAP. Cada vez que se requiera o sea necesario realizar un cambio sobre algún cubo, se tendrá que re-calcular totalmente, para que se reflejen las modificaciones (Chaudhuri, y otros, 2011). La tecnología Palo fue adoptada para el trabajo del Complemento ProCube[16] con almacenes de datos principalmente porque se consideró que las perspectivas de estas tecnologías tienen un amplio potencial. Con la depreciación de la

[16]Complemento creado por Tatiana Mamaliga

memoria computacional y el aumento de la capacidad de la misma, existen posibilidades reales de que sistemas con la tecnología “en memoria” sean más utilizados (Mamaliga, 2013).

1.8.7. Visual Paradigm

Visual Paradigm es una herramienta ideada para soportar el ciclo de vida del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Es útil durante el proceso de análisis y diseño por su probada utilidad para el analista. Este sistema puede manejar gran parte de los diagramas definidos en el lenguaje UML, ya sea creándolos manualmente o importándolos a partir de código en C++, Java, Python, Pascal, Delphi, entre otros, lo cual ofrece posibilidades de ingeniería inversa. Permite crear un modelo y generar el código automáticamente en varios lenguajes para ayudar a comenzar la implementación del proyecto. El código generado incluye definiciones de clases con sus métodos y atributos proporcionando rapidez en el proceso inicial de desarrollo y haciéndolo menos propenso a errores (León, 2013).

1.8.8. JUnit

JUnit es un marco de trabajo para realizar y automatizar pruebas unitarias para el lenguaje de programación Java. Es decir, se sitúa en la fase de pruebas dentro del ciclo de ingeniería del software.

Algunas características de JUnit incluyen (Mañas, 2013):

- Afirmaciones para probar resultados esperados.
- Aparatos de prueba para compartir datos comunes de prueba.
- Corredores de pruebas para pruebas de ejecución.
- JUnit está enlazado como un JAR en tiempo de compilación; el marco de trabajo reside bajo los paquetes: JUnit marco de trabajo, para JUnit 3.8 y anteriores y bajo org.junit para JUnit 4 en adelante.

1.8.9. Metodología de desarrollo AUP

AUP[17]es una versión del Proceso Unificado de Rational (*Rational Unified Process*, RUP) que describe un enfoque sencillo para desarrollar aplicaciones de negocio usando técnicas ágiles y conceptos tomados de RUP. Entre las técnicas agiles que usa se encuentran TDD[18], AMDD[19], la Gestión de Cambios Ágiles y Remodelación de Bases de datos para mejorar la productividad.

Algunas características de esta metodología se enumeran a continuación (ambysoft, 2015):

[17] AUP: Agile Unified Process o Proceso Unificado Ágil

[18]Test Driven Development

[19]Agile Model Driven Development

1. Iterativo e Incremental

   - Se descomponen proyectos grandes en mini-proyectos.
   - Cada mini-proyecto es una iteración.
   - Las iteraciones deben estar controladas.
   - Cada iteración trata un conjunto de casos de uso.

El enfoque iterativo de AUP tiene ciertas ventajas como pueden ser:

- Detección temprana de riesgos.
- Administración adecuada del cambio.
- Mayor grado de reutilización.
- Mayor experiencia para el grupo de desarrollo.

2. Dirigido por Casos de Uso
   - Se centra en la funcionalidad que el sistema debe poseer para satisfacer las necesidades de un usuario (persona, sistema externo, dispositivo) que interactúa con él.
   - Los casos de uso son el hilo conductor que orienta las actividades de Desarrollo.
3. Centrado en la Arquitectura
   Antes que nada se elabora una arquitectura del software que determine la forma que va a tener el mismo.

Además, el ciclo de vida de AUPestá compuesto por las siguientes fases (Flores, y otros, 2014):

- **Inicio**: su objetivo es identificar el alcance inicial del proyecto, proveer una arquitectura potencial para el sistema y obtener un financiamiento inicial del proyecto y la aceptación de los stakeholders[20]. Esta fase está compuesta de las siguientes actividades: especificación de los criterios de éxito del proyecto, definición de los requisitos, estimación de los recursos necesarios ycronograma inicial de fases.

- **Elaboración**: en esta fase se busca probar la arquitectura del sistema y hacer un prototipo de arquitectura que elimine los riesgos técnicos para probar que el proyecto es factible. Las actividades presentes en esta fase son: análisis del dominio del problema, definición de la arquitectura básica, análisis de riesgos, planificación del proyecto.

- **Construcción**: el objetivo es “construir” aplicaciones de forma regular e incremental, que funcionen y satisfagan las necesidades de mayor prioridad de los stakeholders del proyecto. Sus actividades son: análisis, diseño, implementación (codificación), pruebas.

[20]Asociados directa o indirectamente al proyecto

- **Transición**: finalmente se intenta validar e instalar el sistema en el ambiente de producción.

Otro de los aspectos importantes de esta metodología son las disciplinas que definen las actividades a realizar por los miembros del equipo de desarrollo para crear, validar, y entregar la aplicación de tal forma que cumpla con las exigencias de los stakeholders. A continuación se muestran las disciplinas con una pequeña descripción de cada una (Flores, y otros, 2014):

- **Modelado (*Modeling*):** el objetivo de esta disciplina es entender el negocio de la organización.
- **Implementación (*Implementation*):** en esta disciplina se transforma el (los) modelo (s) en código ejecutable y se realiza un nivel básico de pruebas.
- **Prueba (*Test*)**: el objetivo de esta disciplina consiste en realizar una evaluación objetiva para garantizar la calidad.
- **Despliegue (*Deployment*):** en esta disciplina se ejecuta el sistema y se pone a disposición de los usuarios finales.
- **Gestión de configuración** (***Configuration Management*)**: el objetivo de esta disciplina es la gestión de acceso a artefactos de su proyecto.
- **Gestión de proyectos** (***Project Management***): aquí se pretende dirigir las actividades que se llevan a cabo en el proyecto.
- **Entorno (*Environment*)**: finalmente en esta disciplina se apoyan el resto de los esfuerzos para garantizar que el proceso sea el adecuado.

Las fases y disciplinas están reflejadas en la siguiente imagen:

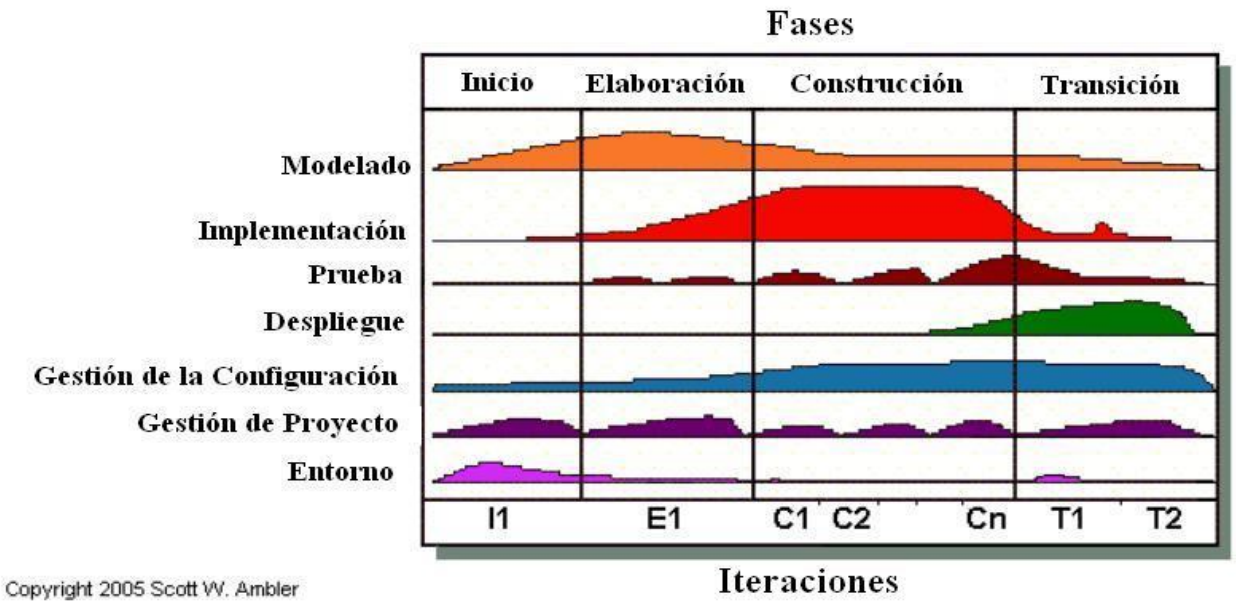

Figura 11 Metodología de desarrollo AUP: elaboración propia

AUP se preocupa especialmente de la gestión de riesgos. Además propone que aquellos elementos con alto riesgo obtengan prioridad en el proceso de desarrollo y sean abordados en etapas tempranas del mismo. También establece un Modelado más simple que el que aparece en RUP por

lo que reúne en una única disciplina las disciplinas de Modelado de Negocio, Requisitos y Análisis y Diseño. El resto de disciplinas coinciden con las restantes de RUP.

1.9. Conclusiones parciales

En este capítulo se abordaron los principales conceptos relacionados con la Minería de Procesos y los Cubos de Procesos. A partir de esta base se realizó un estudio, desde los materiales y documentos disponibles sobre el tema, sobre cómo se vincularon el paradigma Cubos OLAP con la mencionada MP y de cómo esta nueva perspectiva ayuda a resolver problemas existentes de la misma. El estudio de técnicas existentes de Minería de Procesos reveló que los Cubos de Procesos son una vía para mitigar las limitaciones en el diagnóstico de los procesos.

También se realizó una fundamentación teórica de las principales herramientas usadas para llevar a cabo la solución del problema planteado en la Introducción y se concluyó que contribuyen al desarrollo de dicha solución por las características particulares que presentan y que se justifican en este capítulo. Estas herramientas son: MySql y Palo para la gestión de base de datos, Eclipse como entorno de desarrollo integrado, Java como lenguaje de programación y Visual Paradigm como herramienta CASE para el modelado de la solución.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

2.1. Introducción

En este capítulo se describe el desarrollo de la solución propuesta mediante los artefactos que propone la metodología AUP. Para ello se capturaron y especificaron los requisitos del sistema, se definieron los casos de uso representados en el diagrama de casos de uso del sistema. Además, se describe la aplicación de la arquitectura de tres capas. Durante el análisis y diseño del sistema se generaron diagramas como el de colaboración y el de clases del diseño. Se especifican los paquetes en los que se divide la aplicación y los patrones de asignación de responsabilidades utilizados, así como los estándares de codificación empleados.

2.2. Planificación del proyecto

A continuación se muestra una planificación del Proyecto desarrollado, teniendo en cuenta las actividades a realizar en cada una de las fases, el tiempo asignado la fase y los recursos asociados a ella.

Tabla 1 Planificación de cada una de las fases de AUP

Fase	Actividades	Duración	Recursos
Inicio	En esta fase se hace un análisis general del sistema que contiene: captura de requisitos, descripción del sistema y casos de uso.	2 Semanas	1 Computadora y 2 Personas.
Elaboración	En esta fase se hace un diseño del sistema compuesto por: arquitectura de software e interfaces de usuario.	2 Semanas	1 Computadora y 2 Personas.
Construcción	En esta fase se hace la construcción de la extensión del Complemento, así como el análisis y diseño del mismo, esto comprende: los estándares de codificación, la realización de los casos de uso, el diagrama de	4 Semanas	1 Computadora y 2 Personas.

	clases, el modelo de despliegue y el código fuente.		
Transición	En esta fase se prueba y valida la solución propuesta para favorecer el despliegue.	4 Semanas	1 Computadora y 2 Personas.

2.2.1. Diagrama de Gantt

Las siguientes imágenes muestran cómo se hizo la planificación utilizando la técnica de diagrama de Gantt. En la primera imagen se puede observar las actividades a realizar y el tiempo de duración de las mismas. Por otro lado en la segunda imagen se representan las actividades en un calendario donde se puede observar la relación entre ellas:

Nombre	Duracion	Inicio	Terminado	Predecesores
Captura de Requisitos	3 days	2/02/15 8:00	4/02/15 17:00	
Descripción del Sistem	2 days?	5/02/15 8:00	6/02/15 17:00	1
Generación de Casos	2 days?	9/02/15 8:00	10/02/15 17:00	2
Gestión de Riesgos	3 days?	11/02/15 8:00	13/02/15 17:00	3
Análisis de la Arquitec	5 days?	16/02/15 8:00	20/02/15 17:00	4
Elaboración de los Prc	5 days?	23/02/15 8:00	27/02/15 17:00	5
Elaboración de los Est	1 day?	2/03/15 8:00	2/03/15 17:00	6
Descripción de los Cas	4 days?	3/03/15 8:00	6/03/15 17:00	7
Análisis y Diseño del S	10 days?	9/03/15 8:00	20/03/15 17:00	8
Implementación del Si	24 days?	23/03/15 8:00	23/04/15 17:00	9
Validación y Prueba de	15 days?	13/04/15 8:00	1/05/15 17:00	9

Figura 12 Actividades llevadas a cabo en el Capítulo 2 y 3: elaboración propia.

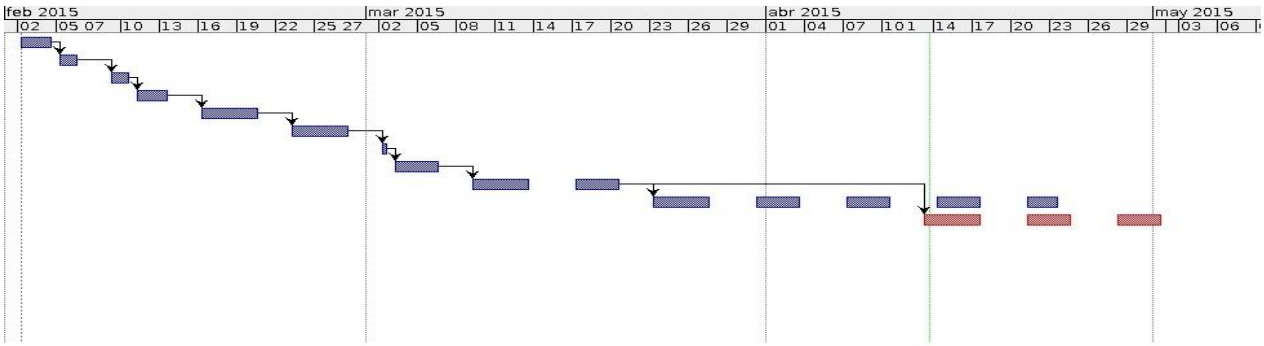

Figura 13 Diagrama de Gantt con las actividades vistas anteriormente: elaboración propia.

2.3. Fase de Inicio

Como se ha dicho, en la primera fase (Inicio) es donde se generan artefactos como: alcance del proyecto, requisitos funcionales y no funcionales, análisis del sistema, gestión de riesgos y casos de uso, de tal forma que al finalizar la fase el equipo de trabajo y los stakeholders sean capaces de comprender el sistema a fondo, así como su envergadura, de lo contrario el proyecto debe ser redirigido o cancelado.

2.3.1. Descripción del sistema

El sistema es un una extensión del Complemento ProCube para ProM, que es una aplicación de escritorio destinada al descubrimiento y análisis de modelos de procesos. Este Complemento muestra un Cubo de Procesos conformado a partir de un registro de eventos y permite la aplicación de diversas operaciones OLAP como *Slice*, *Dice*, *Roll-Up* y *Drill-Down* para obtener modelos de procesos de las diferentes celdas del cubo. Además muestra con valores numéricos un análisis comparativo entre las diferentes celdas.

2.3.2. Modelo conceptual

El modelo conceptual explica cuáles son y cómo se relacionan entre sí los conceptos relevantes en un problema determinado. En la siguiente figura se muestra el modelo conceptual referente a la presente investigación:

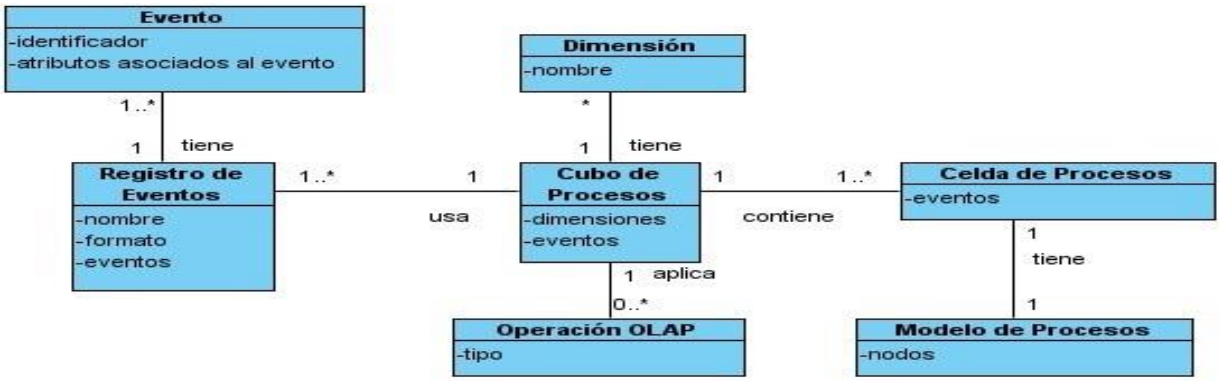

Figura 14 Modelo conceptual referente a la presente investigación: elaboración propia

Para una mejor comprensión, a continuación se describe cada uno de los conceptos que intervienen en el Modelo Conceptual:

Cubo de Procesos: representa el concepto de Cubos de Procesos que posee dimensiones y está compuesto por celdas de procesos. Se forma a partir de un registro de eventos y puede aplicar operaciones OLAP para modificar su estructura.

Celda de Procesos: es parte de un Cubo de Procesos y en su interior tiene eventos que son un subconjunto del registro de eventos. Cada celda tiene asociado un modelo de procesos.

Registro de eventos: contiene un conjunto de eventos que están presentes en la ejecución de un proceso. Puede tener diferentes formatos: xes, mxml, cvs, entre otros. Se usa para crear los Cubos de Procesos.

Evento: es una instancia de una actividad, o sea una de las formas en que puede realizarse una actividad dentro de un proceso determinado. Es la base del registro de eventos y tiene asociado un conjunto de atributos entre los que pueden estar: recurso, tiempo o costo.

Dimensión: es un atributo que representa una de las dimensiones del cubo.

Operación OLAP: representa a las operaciones que se pueden realizar sobre el cubo. El tipo de operación puede ser: *Dice, Slice, Drill-Down y Roll-Up*.

Modelo de procesos: es la representación en forma de modelo de un proceso determinado. Contiene nodos asociados a las actividades.

2.3.3. Técnicas de captura de requisitos

Existen varias técnicas de captura de requisitos, para las cuales se debe tomar en cuenta las características propias del proyecto en particular que se esté desarrollando. Esto permite obtener de forma clara las necesidades planteadas por los clientes y así lograr la satisfacción de estos. Las técnicas a utilizar para la captura de requisitos son (Mendez, 2010):

- Entrevistas: es una de las técnicas más usadas en la captura de requisitos. Consiste en establecer una conversación entre personas de ambas partes. Estas son aplicadas a los especialistas funcionales. Pueden ser lo mismo provechosos o no, ya que dependen de las habilidades del entrevistador y los entrevistados para obtener información con la mayor calidad posible
- Tormenta de ideas: reunión de varios interesados en la que todos expresan sus ideas sobre el problema y su solución. La forma de llevarla a cabo es que cada participante diga su idea sin ser interrumpido por otro. Al finalizar la sesión de lluvia de ideas se puede hacer una recolección de ideas sin duplicidad.
- Casos de uso / Escenarios: requisitos en contexto de uso.
- Sistemas existentes: utilizada durante el análisis de varios sistemas existentes que estén relacionados con el que va hacer construido. Analizándolos en cuanto a un grupo de características necesarias que debe llevar el que se va a elaborar.
- Mapas conceptuales: es otra técnica bastante común, usada para la representación gráfica de las ideas y sus relaciones.

Para la captura de los requisitos que guiaron el desarrollo de la herramienta en la presente investigación se hizo uso de las técnicas entrevistas, tormenta de ideas, casos de uso y modelos conceptuales.

2.3.4. Requisitos funcionales

Los requisitos funcionales (RF) son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. Además, estos describen lo que el sistema debe hacer. Estos requisitos dependen del tipo de software que se desarrolle y de los posibles usuarios de dicho software. Son una descripción de cómo se debe comportar el sistema, de un atributo o una propiedad. Además constituyen la automatización de los casos de usos (Larman, 2010). A continuación se describen los requisitos funcionales que debe poseer la extensión a desarrollar:

RF1 Importar registros de eventos: el sistema debe importar registros de eventos en el formato .xes que sirven como entrada al Complemento. Este Requisito posee una complejidad de implementación **media** y una prioridad **alta.**

RF2 Convertir registro de eventos en Cubo de Procesos: el sistema debe convertir el registro de eventos en un Cubo de Procesos mediante la conexión del Complemento con el software PALO. Posee una complejidad de implementación **alta** y una prioridad **alta.**

RF3 Filtrar dimensiones del CP: el sistema debe filtrar un grupo de atributos que serán tomados como las dimensiones del cubo. Posee una complejidad **alta** y una prioridad también **alta**.

RF4 Aplicar las operaciones *Slice* y *Dice*: el sistema debe aplicar las operaciones *Slice* y *Dice* sobre el Cubo de Procesos para obtener un nuevo cubo. Tiene una complejidad de implementación **baja** y una prioridad **media**.

RF5 Aplicar las operaciones Drill-Down y Roll-Up: el sistema debe aplicar las operaciones Drill-Down y Roll-Up sobre el cubo para obtener nuevas perspectivas de análisis. Tiene una complejidad de implementación **baja** y una prioridad **media**.

RF6 Comparar celdas del CP: el sistema debe mostrar una comparación entre dos celdas teniendo en cuenta varios indicadores (por ejemplo: tiempo, costo y recurso). Su complejidad de implementación es **alta** y la prioridad es **alta**.

RF7 Visualizar los modelos de las celdas de los CP: el sistema debe visualizar los modelos de procesos de las celdas a comparar en ventanas paralelas usando el algoritmo de Alpha Miner que visualiza el resultado en forma de redes Petri. Su complejidad es **alta** y su prioridad también es **alta**.

2.3.5. Requisitos no funcionales

Los requisitos no funcionales (RNF) son propiedades o cualidades que el sistema debe tener. Son características que hacen al producto (sistema desarrollado) atractivo, usable, rápido o confiable y por lo tanto son fundamentales para el éxito del mismo (Larman, 2010). A continuación se describen los requisitos no funcionales que debe poseer la extensión a desarrollar:

Usabilidad

RNF1: el sistema podrá ser usado por cualquier persona que posea conocimientos básicos en computación. También presenta una interfaz amigable y fácil de entender.

Portabilidad

RNF2: el sistema podrá ser ejecutado sobre los sistemas operativos GNU/Linux o Windows, permitiendo una fácil migración haciendo uso de estándares y tecnologías de código abierto. Tendrá que convertirse en un paquete para el ProM, funcionando así como un Complemento para poder usarse en cualquier estación de trabajo.

Requisitos de software y hardware

RNF3: se pueden utilizar los sistemas operativos Windows, a partir de la versión 7 y Linux en cualquiera de sus distribuciones, se debe tener instalada la máquina virtual de java en su versión 7 y tener algunos paquetes del ProM para realizar la modelación, como el Alpha Miner, Dotted Chart, Fuzzy Miner, entre otros. Además se deben tener instalados Palo y MySql.

RNF4: se debe reservar para el sistema 4 GB de memoria RAM y 2 GB de espacio libre para el disco duro, donde se guardarán los paquetes del ProM.

Soporte

RNF5: la aplicación contará antes de su puesta en marcha con un período de pruebas, se le dará mantenimiento, configuración y se brindará el servicio de instalación.

Reusabilidad

RNF6: el sistema desarrollado debe permitir ser utilizado en trabajos futuros.

Mantenimiento

RNF7: el sistema debe ser fácil de mantener después de desarrollado.

2.3.6. Casos de Usos del Sistema

Los casos de uso del sistema describen las funcionalidades que el software debe tener para poder satisfacer las necesidades del cliente. También de cierta forma debe mostrar una interacción con un usuario o con otros sistemas (Larman, 2010). La siguiente figura muestra el diagrama de casos de uso del presente sistema:

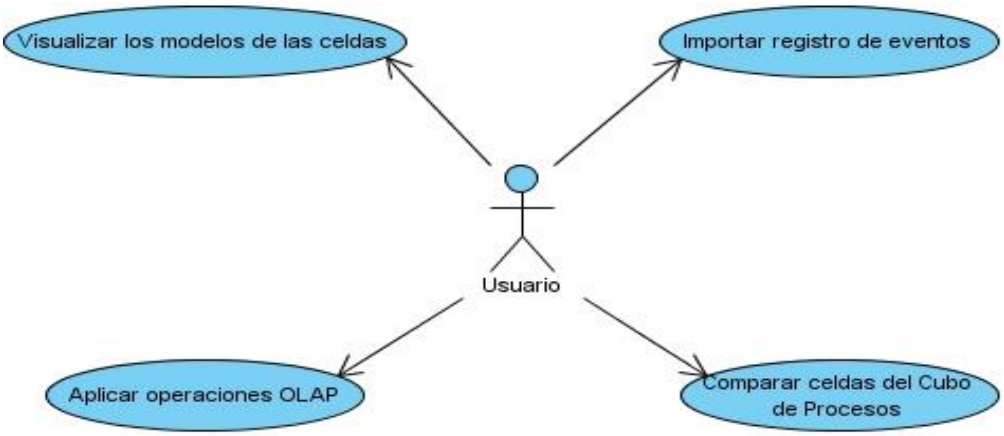

Figura 15 Diagrama de casos de uso del sistema: elaboración propia

2.3.7. Métricas para validar la calidad de los requisitos

Una vez definidos los requisitos del sistema estos deben de ser validados para asegurar que el análisis de los mismos y los resultados obtenidos durante la definición de requisitos son correctos. La validación de requisitos tiene como misión demostrar que la definición de los requisitos especifica realmente el sistema que el usuario necesita o que el cliente desea.

Para ello se examinan las especificaciones para asegurar que todos los requisitos del sistema han sido determinados sin ambigüedad, sin inconsistencias, sin omisiones, que los errores detectados hayan sido corregidos y que el resultado del trabajo se ajusta a los estándares para el proceso, el proyecto y el producto (Fornaris, y otros, 2010).

Para la validación de los requisitos de la solución propuesta se decidió la utilización de tres métricas (Fornaris, y otros, 2010):

- Estabilidad de los requisitos
- Grado de validación de los requisitos
- Especificidad de los requisitos

La **Estabilidad de los requisitos** se mide de la siguiente forma:

ETR = [RT-RM /RT] *100

Donde:

ETR: estabilidad de los requisitos.

RT: total de requisitos definidos.

RM: cantidad de requisitos modificados.

De los siete requisitos funcionales solo uno fue modificado por lo tanto la métrica se aplicó de la siguiente manera:

ETR = [7-1/7] * 100 = 85.71

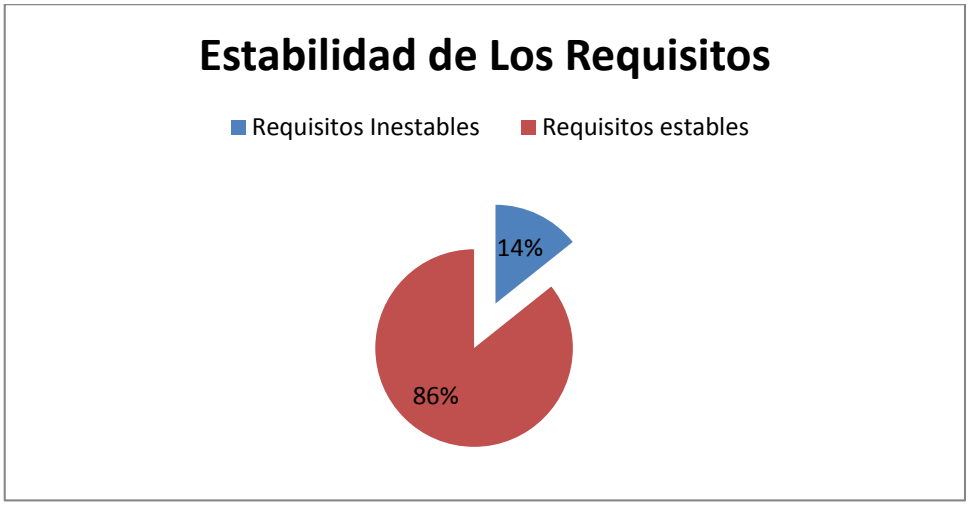

Figura 16 Estabilidad de los requisitos: elaboración propia

Por otra parte la técnica **Grado de validación** se usa para medir que los requisitos identificados con el cliente fueron correctos:

VR = NC/ (NC + NNV)

VR = 7/ (7+0) = 1.

Donde:

VR: grado de validación de los requisitos.

NC: número de requisitos que se han validado como correctos.

NNV: número de requisitos no validados aún.

De acuerdo al resultado obtenido y coincidiendo este con el valor óptimo de esta métrica se asegura que existe un alto nivel de corrección en la definición de los requisitos

Por último para cuantificar la **Especificidad de los requisitos** se aplicó la técnica de la siguiente manera:

Q1 = NUI/NR

Q1 = 7/7 = 1.

Donde:

Q1: especificidad de los requisitos.

NR: total de requisitos definidos.

NUI: cantidad de requisitos para los que los revisores tuvieron interpretaciones idénticas.

Según el resultado de la métrica aplicada los requisitos no tienen ambigüedad y por tanto la especificación de requisitos se hizo con calidad.

2.4. Fase de Elaboración

La fase de Elaboración es donde se realiza un diseño general del sistema, mediante el modelado de la arquitectura de software. Al finalizar la fase debe quedar clara la estrategia para desarrollar la extensión del Complemento.

2.4.1. Arquitectura de Tres Capas

El diseño de la arquitectura de un sistema, es el proceso por el cual se define una solución para los requisitos técnicos y operacionales del mismo. Este proceso define qué componentes conforman el sistema, cómo se relacionan entre ellos y cómo mediante su interacción llevan a cabo la funcionalidad especificada.

Se hará uso de la Arquitectura Física de Tres Capas, que es una generalización de la arquitectura cliente-servidor donde la carga se divide en tres partes con un reparto claro de funciones: una capa de presentación, otra para las reglas del negocio y la capa de datos para el almacenamiento de los mismos, esto facilita la reutilización, debido a que los datos abstractos pueden ser utilizados por diferentes implementaciones de una misma capa en la medida que soporten las mismas interfaces (Pressman, 2003). A continuación se describen cada una de las tres capas:

- **Capa de presentación**: presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). También es conocida como interfaz gráfica y debe tener la característica de ser "amigable" (entendible y fácil de usar) para el usuario. Esta capa se comunica únicamente con la capa de negocio. Para el Complemento presentado en este trabajo de diploma la capa de presentación la brinda el marco de trabajo ProM.
- **Capa de negocio**: es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta envuelve el software de aplicación usado en la programación de la solución. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. En este caso particular la capa de negocio está constituida por el propio Complemento.
- **Capa de datos**: es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio. En esta capa se encuentran los registros de eventos manipulados por el Complemento.

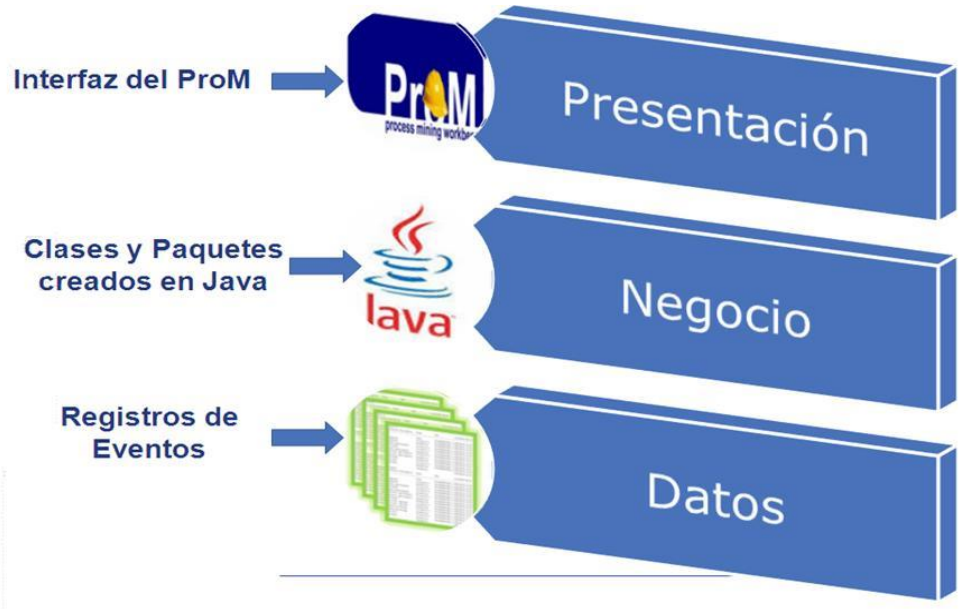

Figura 17 Arquitectura de tres capas: elaboración propia

2.5. Fase de Construcción

La fase Construcción es donde se generan los artefactos relacionados con la implementación del sistema, entre ellos están la realización de los casos de uso, el modelo de despliegue, los diagramas de clases tanto de análisis como de diseño y por supuesto el propio sistema.

2.5.1. Análisis

Como se conoce, en AUP las actividades se repitan de una fase a otra. Así pues, aunque el punto fuerte del análisis y diseño sean las fases de Inicio y Elaboración, en la fase de Construcción se realizan estas actividades (Análisis, Diseño, Implementación y Prueba). En el análisis los modelos son conceptuales, o sea son una abstracción del sistema, de la misma forma existen tres tipos de clases conceptuales, ellas son: clases de control, de interfaz y de entidades.

2.5.1.1. Estándares de codificación

Los estándares de codificación que a continuación se definen tienen como objetivos aumentar la portabilidad, reducir el esfuerzo de mantenimiento y sobre todo, mejorar la legibilidad del código desarrollado en Java.

Se nombrarán los ficheros .java usando la notación CamelCase:

NombreDeLaClase.java

En los **diagramas de clases de análisis** se representarán las clases de la siguiente manera:

- Las **clases entidad** se nombrarán siguiendo el siguiente patrón:

CE_NombreDeLaClase

- Las **clases controladoras** se nombrarán siguiendo el siguiente patrón:

CC_NombreDeLaClase

- Las **clases interfaces** se nombrarán siguiendo el siguiente patrón:

CI_NombreDeLaClase

- **Variables**:

Los nombres de las variables comenzarán con un identificador del tipo de dato al que correspondan y comenzando en minúscula como se muestra a continuación:

/* tipos simples */

Bool bVarName;

Int iName;

Float fName;

- **Nomencladores:**

Todos los nomencladores están escritos en inglés por convención de la comunidad de Minería de Procesos para facilitar la distribución y el entendimiento de las aplicaciones que se hagan dentro de

esta. Además, las clases comenzarán con letra inicial mayúscula y las variables y métodos con minúscula.

2.5.1.2. Descripción del Caso de Uso Comparar celdas de un CP

La descripción del caso de uso, como su nombre lo indica, describe la interacción del usuario con el sistema y las respuestas que este último brinda.

A continuación se muestra la descripción del caso de uso Comparar celdas de un Cubo de Procesos:

Tabla 2 Descripción del CU Comparar celdas de un CP

Nombre del caso de uso	Comparar celdas de un Cubo de Procesos
Actor	Usuario (Inicia)
Propósito	Comparar dos celdas del Cubo de Procesos
Descripción	El caso de uso comienza una vez el usuario elige los atributos que servirán de dimensiones, después despliega el menú de operaciones del Cubo de Procesos y elige la opción “Comparar”, luego el sistema muestra una ventana para configurar la comparación.
Precondiciones	Para realizar la comparación se debe haber importado un registro de eventos.
PostCondiciones	Luego de comparar se muestra una ventana con las diferencias entre las celdas
Curso Normal de los Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El usuario filtra los atributos que serán dimensiones del cubo.	1.1. El sistema habilita una pestaña donde se representa en forma de tabla el cubo.
2. El usuario despliega el menú de operaciones y selecciona la opción comparar.	2.1. El sistema muestra una ventana donde se configura la comparación, además de los modelos de procesos.
3. El usuario selecciona los atributos, los indicadores de los atributos y los procesos que desea comparar.	3.1. El sistema muestra una interfaz con la comparación teniendo en cuenta el indicador seleccionado por el usuario.

4. El usuario selecciona la opción de visualizar la comparación en forma de gráfico de barras.	4.1. El sistema visualiza el resultado.
Curso Alterno	
3. El usuario selecciona un atributo que no tiene comparación numérica.	3.1. El sistema muestra un mensaje de que no hay comparación posible a realizar.
Prioridad	Crítico.

2.5.2. Diseño

A diferencia del Análisis, el Diseño es un modelo físico para la implementación de un sistema, o sea esta disciplina se lleva a cabo teniendo en cuenta los objetos reales de dicho sistema y no abstracciones del mismo. El objetivo de esta disciplina es crear una entrada apropiada y un punto de partida para la implementación, así como descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo.

2.5.2.1. Organización en paquetes

Los paquetes son un mecanismo de organización de elementos que subdividen el modelo en otros más pequeños que colaboran entre sí. Tienen que ser (Larman, 2010):

- **Cohesivos**: sus contenidos deben estar fuertemente relacionados.
- **Débilmente acoplados**: minimizar las dependencias entre paquetes.

En este caso particular el diagrama de paquetes está compuesto por cuatro paquetes. El paquete “operation” contiene la lógica del negocio relacionada a la comparación en el Cubo de Procesos. El paquete “procube” contiene la lógica del negocio que crea y maneja los Cubos de Procesos. El paquete “processmining” contiene los Complementos del ProM. Finalmente el paquete “tensegrity” maneja la seguridad del Complemento.

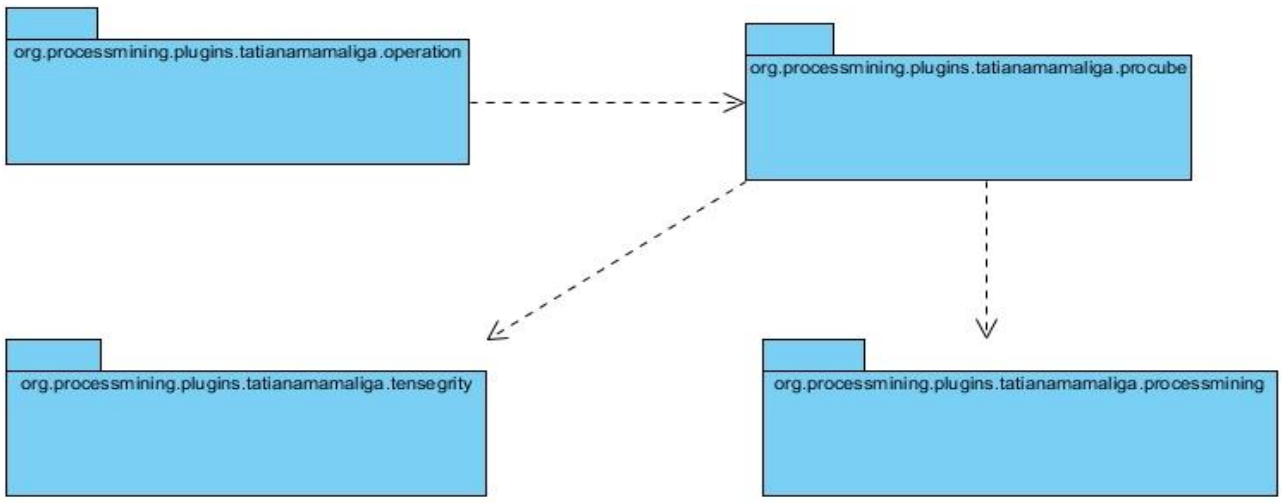

Figura 18 Organización en paquetes: elaboración propia

2.5.2.2. Diagrama de Clases del Diseño del paquete operation

El diagrama de clases del diseño es una representación de las clases reales que componen el negocio, así como las relaciones que existen entre ellas. En el anexo 3 se muestra el diagrama de clases del paquete “operation”:

2.5.2.3. Patrones de asignación de responsabilidades

UML define una responsabilidad como "un contrato u obligación de un clasificador". Las responsabilidades están relacionadas con las obligaciones de un objeto en cuanto a su comportamiento. Estas responsabilidades son de los siguientes dos tipos (Larman, 2010):

- Conocer.
- Hacer.

Un patrón es una descripción de un problema y la solución, a la que se da un nombre y que se puede aplicar a nuevos contextos; idealmente, proporciona consejos sobre el modo de aplicarlo en varias circunstancias y considera los puntos fuertes y compromisos. Muchos patrones proporcionan guías sobre el modo en el que deberían asignarse las responsabilidades a los objetos, dada una categoría específica del problema (Larman, 2010).

Los principales patrones de asignación de responsabilidades (GRASP) son (Sommerville, 2005):

- Experto en Información.
- Creador.
- Alta Cohesión.
- Bajo Acoplamiento.
- Controlador.

El patrón **Experto** es el encargado de asignar aquellos objetos con la responsabilidad de conocer lo necesario para resolver un problema, es por eso que se le denomina **Experto en Información**, pues el objeto con esa responsabilidad conoce todo lo relacionado a los otros objetos. En este caso particular esa responsabilidad la puede tener la clase CC_CompararCeldas, quien obtiene toda la información necesaria para ejecutar las acciones de los otros objetos.

Debido a que a partir de un registro de eventos se crean los Cubos de Procesos, se le asigna a la clase CE_RegistroEventos la responsabilidad de crear objetos mediante el patrón **Creador.**

Los patrones de **Bajo Acoplamiento** y la **Alta Cohesión** se ven reflejados en los diagramas de paquetes. El primero se refiere a la poca dependencia con que deben ser diseñadas las clases del sistema y el segundo a la gran relación que deben tener todos los métodos de una misma clase, de forma tal que dicha clase no maneje objetos totalmente diferentes entre sí. El uso correcto de estos dos patrones contribuye a la reutilización del código.

Por último se encuentra el patrón **Controlador** encargado de controlar la mayoría de las tareas del sistema, en este caso la clase con esta responsabilidad es CC_CompararCeldas.

2.5.2.4. Modelo de despliegue

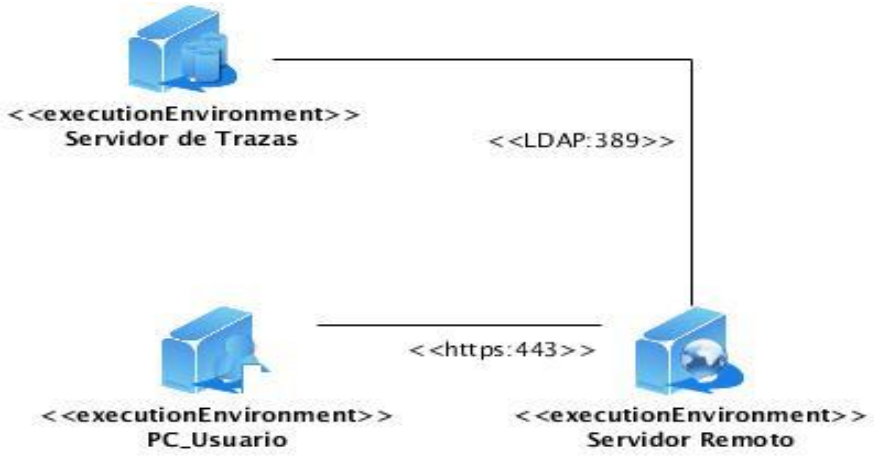

Figura 19 Modelo de despliegue: elaboración propia

2.5.3. Implementación

El objetivo de esta disciplina es transformar el diseño en código ejecutable mediante la implementación de la lógica del negocio. Además, esta disciplina permite crear las interfaces de usuario en pos de mejorar la interacción con los usuarios.

A continuación se muestran las principales interfaces del sistema desarrollado. En la primera imagen se puede apreciar la interfaz del ProM donde se seleccionan los Complementos a utilizar:

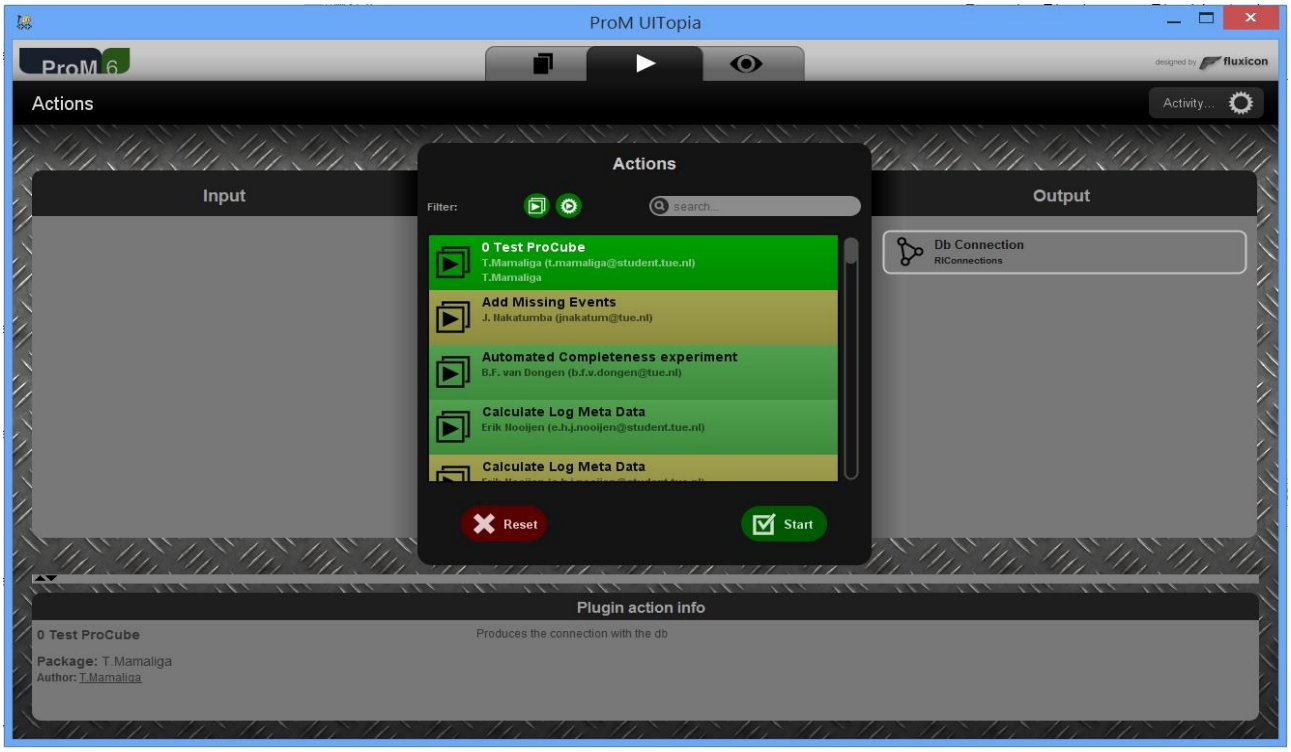

Figura 20 Interfaz de ProM para la selección de los Complementos: elaboración propia.

Luego de seleccionar el Complemento a utilizar se muestra la ventana principal de dicho Complemento, donde se manipulan las bases de datos y dimensiones. Esta interfaz responde a los requisitos funcionales dos y tres:

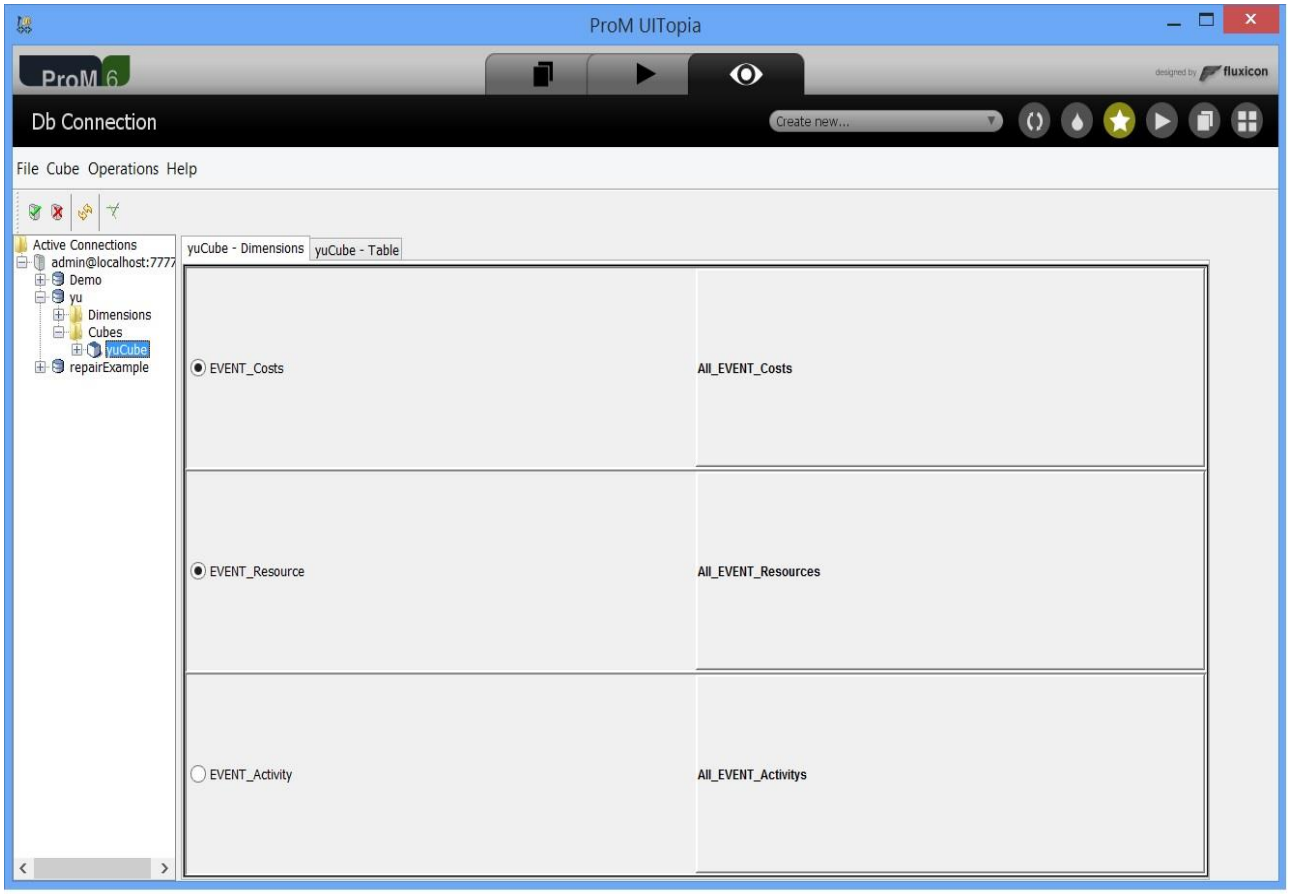

Figura 21 Interfaz para conformar el CP: elaboración propia

En la siguiente imagen se puede observar la ventana de comparación, a la izquierda de la imagen se encuentran los atributos por los que comparar y a la derecha las celdas que se desean comparar. Esta interfaz responde a los requisitos funcionales cuatro, cinco y seis**:**

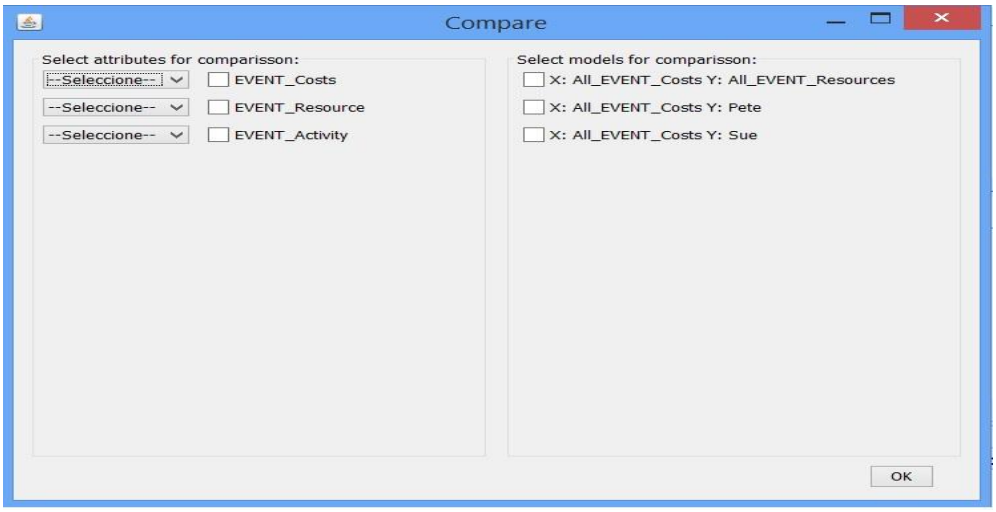

Figura 22 Interfaz para seleccionar los parámetros de comparación: elaboración propia

Las dos imágenes que le siguen corresponden a los otros tipos de comparación que son posibles realizar, es decir a través de tablas estadísticas y de los modelos. Son representaciones de los requisitos funcionales seis y siete respectivamente:

Models Data

EVENT_amount

Activity	Sum		Average		Median	
	Log1	Log2	Log1	Log2	Log1	Log2
Send Fine	-	0.0	-	0.0	-	0.0
Create Fine	880.0	-	22.0	-	22.0	-
Add penalty	25579.0	286.0	68.03	71.5	71.5	71.5
Insert Fine Notification	0.0	0.0	0.0	0.0	0.0	0.0
Send for Credit Collection	-	-	-	-	-	-
Payment	0.0	0.0	0.0	0.0	0.0	0.0
Insert Date Appeal to Prefecture	0.0	0.0	0.0	0.0	0.0	0.0
Send Appeal to Prefecture	0.0	0.0	0.0	0.0	0.0	0.0
Appeal to Judge	0.0	0.0	0.0	0.0	0.0	0.0
Notify Result Appeal to Offender	-	0.0	-	0.0	-	0.0
Receive Result Appeal from Prefecture	-	0.0	-	0.0	-	0.0
Total	26459.0	286.0	-	-	-	-

Figura 23 Interfaz donde se muestra el resultado de la comparación: elaboración propia.

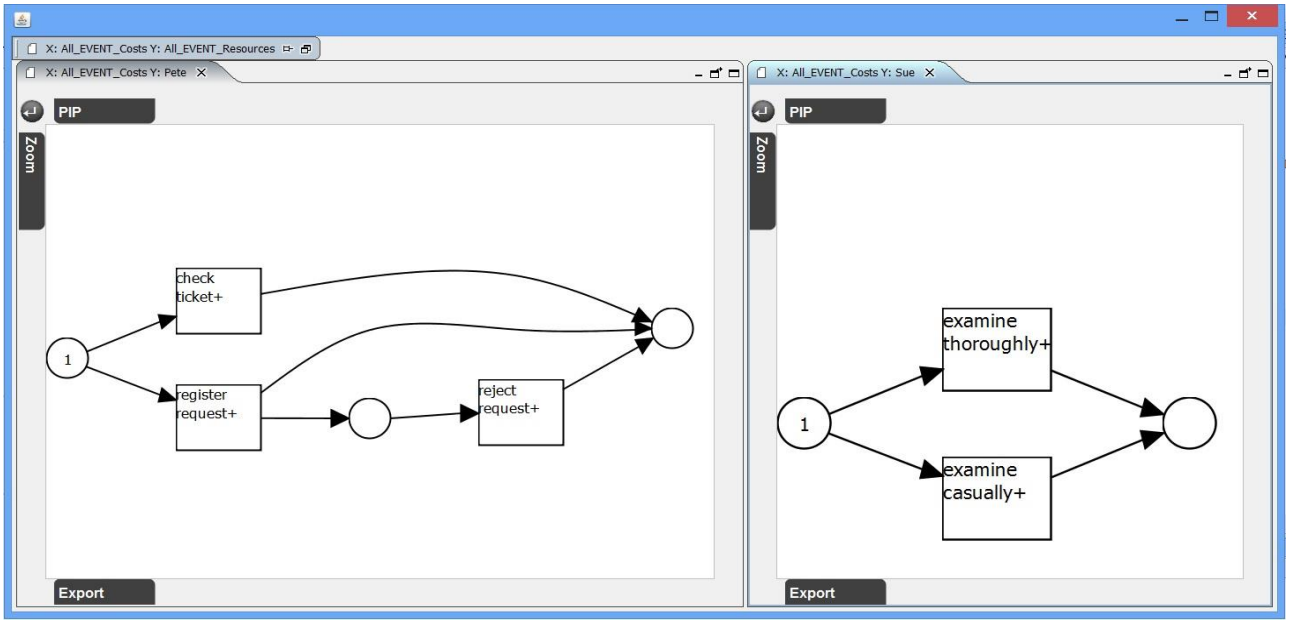

Figura 24 Comparación a nivel de modelos: elaboración propia

2.6. Conclusiones parciales

En este capítulo se muestra cómo se desarrolló una solución siguiendo las fases de la metodología AUP, dicha solución constituye una extensión de un Complemento para el marco de trabajo ProM. Al usar AUP se generaron artefactos propios de esta metodología que facilitaron el desarrollo de la solución. La arquitectura de tres capas contribuyó a distribuir las funciones en las diferentes capas de forma tal que contribuyan entre sí sin que exista interferencia entre las mismas. Para llevar a cabo la implementación se tuvo en cuenta los estándares de codificación presentados en el documento y la utilización de patrones de asignación de responsabilidad que facilitaron el desarrollo de la extensión.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN

3.1. Introducción

En el presente capítulo se realizan las pruebas para validar la solución en pos de garantizar que el sistema pueda ser desplegado satisfactoriamente. Además se valida que la solución cumpla con las expectativas del cliente y se comparan los resultados obtenidos al aplicar el complemento extendido al registro de eventos del departamento de control de tráfico de Italia, con los resultados obtenidos de aplicar otras técnicas de MP. Esto se hace para validar que la solución resuelve el problema planteado al inicio de la investigación.

3.2. Pruebas de software

Las pruebas presentan una interesante anomalía para el ingeniero del software. Durante las fases anteriores de definición y de desarrollo, el ingeniero intenta construir el software partiendo de un concepto abstracto y llegando a una implementación tangible. A continuación, se realizan las pruebas. El ingeniero crea una serie de casos de prueba que intentan destruir el software construido. De hecho, las pruebas son uno de los pasos de la ingeniería del software que se puede ver como destructivo en lugar de constructivo (Pressman, 2003).

3.3. Nivel de unidad

A nivel de unidad se realizan pruebas de funcionalidad para probar el correcto funcionamiento de un módulo. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. Estas pruebas de funcionalidad se clasifican en pruebas de caja blanca y pruebas de caja negra.

3.3.1. Prueba de Caja Blanca

La prueba de caja blanca, denominada a veces prueba de caja de cristal, es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo; se ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa; se ejecuten todos los bucles en sus límites y con sus límites operacionales; y se ejerciten las estructuras internas de datos para asegurar su validez (Pressman, 2003).

Según (Patton, 2005) existen dos formas de realizar las pruebas de caja blanca:

Pruebas estáticas: es el proceso que cuidadosa y metódicamente revisa el diseño del software, la arquitectura o el código para encontrar defectos sin necesidad de ejecutar el código. Esto algunas veces se refiere a un análisis estructural.

Pruebas dinámicas: en estas pruebas se revisa dentro de la "caja", se examina el código mientras se ejecuta. Esta prueba utiliza la información que se obtiene al observar cómo funciona el código para determinar qué y cómo probar, además de cómo aproximarse a las pruebas.

3.3.1.1. Prueba del camino básico

El método del camino básico permite obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida corno guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecute por lo menos una vez cada sentencia del programa (Myers Glenford, y otros, 2011). Este método pertenece a la clasificación de pruebas estáticas.

Para realizar el método antes mencionado es necesario hacer una representación del código en forma de grafos, el grafo correspondiente a cada estructura se muestra en la siguiente imagen[21]:

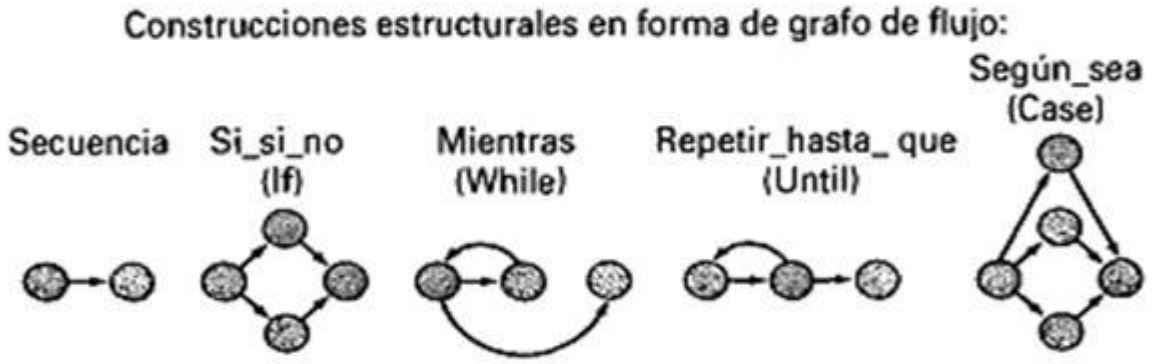

Figura 25 Representación en grafos de las estructuras del código

Otro aspecto importante para tratar con grafos de flujo es la complejidad ciclomática. La complejidad ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto del método de prueba del camino básico, el valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y nos da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez (Myers Glenford, y otros, 2011). Por otra parte los mencionados caminos independientes indican las posibles ejecuciones de un procedimiento en sus versiones verdaderas y falsas, o sea permite conocer todas las posibilidades de ejecución y a partir de aquí se pueden diseñar casos de prueba que lleven a cabo todas estas ejecuciones para conocer sus respuestas.

La complejidad ciclomática para descubrir la cantidad de caminos independientes se puede calcular de tres formas (Pressman, 2003):

1. El número de regiones del grafo de flujo coincide con la complejidad ciclomática.

[21] Imagen tomada de (Pressman, 2003)

2. La complejidad ciclomática, V(G), de un grafo de flujo G se define como:

V (G)=A-N+2

Donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.

3. La complejidad ciclomática, V(G), de un grafo de flujo G también se define como:

V(G) =P+ 1

Donde P es el número de nodos predicado[22]contenido en el grafo de flujo G.

Si se toma como ejemplo la siguiente imagen[23], se llega a la conclusión que la complejidad ciclomática es cuatro y por lo tanto existen cuatro caminos, ellos son:

Camino 1: 1-11

Camino 2: 1-2-3-4-5-10-1-11

Camino 3: 1-2-3-6-8-9-10-1-1 1

Camino 4: 1-2-3-6-7-9- 10-1-1 1

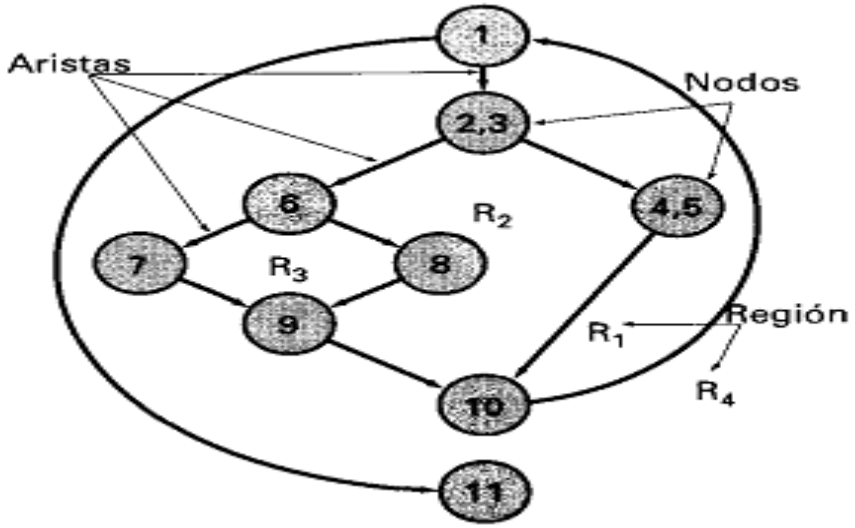

Figura 26 Representación en forma de grafo para determinar la complejidad ciclomática

A continuación se muestra como se aplicó el método anterior en el presente trabajo de diploma. La imagen muestra un método de la clase CompareController y cada nodo del grafo está delimitado por un rectángulo:

[22]Nodo predicado: aquel nodo de donde salen dos aristas.

[23]Imagen tomada de (Pressman, 2003)

```
private List<List<Map<String, String>>> getDimElements() {
List<String> emptyNets = new ArrayList<String>();
final List<List<Map<String, String>>> dimElementList = new ArrayList<List<Map<String,
String>>>();
ProCubeBrowserTable table = pCUI.getBrowser().getProCubeBrowserTable();
DefaultTableModel model = (DefaultTableModel) table.getModel();                    1
PaloBrowser browser = pCUI.getBrowser();
for (int i = 0; i < table.getRowCount(); i++) {                                    2
    for (int j = 1; j < table.getColumnCount(); j++) {                             3
        String colName = table.getColumnModel().getColumn(j).getHeaderValue().toString();
        String rowName = model.getValueAt(i, 0).toString();
        String name = "X: " + rowName + " Y: " + colName;
        List<Element[]> filterElement = browser.getAllFiltersElements(rowName, colName);
        List<Map<String, String>> DimElement = operation.EventsFromDimension(browser.
        getDbName(),                                                               4
             filterElement, pCUI.getIdbConn(), pCUI.getRdbConn());
        if (!DimElement.isEmpty()) {                                               5
            netNames.add(name);                                                    6
            dimElementList.add(DimElement);
        } else
            emptyNets.add(name);                                                   7
    }
    }
    if (!emptyNets.isEmpty())                                                      8
        JOptionPane.showMessageDialog(null, "Some models are empty:" + emptyNets.
        toString(), "Empty models", 1);                                            9
    return dimElementList;                                                         10
}
```

Figura 27 Método getDimElements() de la clase CompareController: elaboración propia

Luego el grafo correspondiente sería:

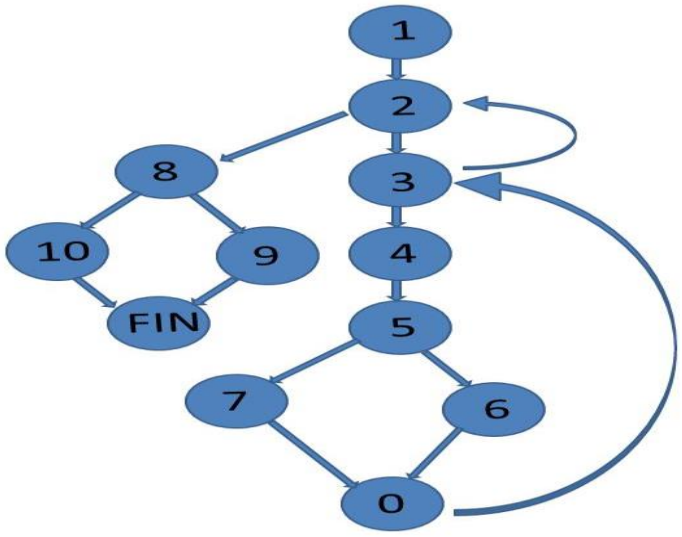

Figura 28 Grafo correspondiente al método getDimElements(): elaboración propia

Donde el nodo cero es auxiliar para denotar el fin del blucle **for** representado por el nodo tres y de igual manera el nodo FIN representa el final del método. De esta forma la complejidad ciclomática está dada por:

V (G)=A-N+2

V (G)=15-12+2=5

Por lo tanto existen como máximo cinco caminos independientes, estos son:

Camino 1: 1-2-8-9-FIN

Camino 2:1-2-8-10-FIN

Camino 3:1-2-3-2-8-9-FIN

Camino 4:1-2-3-4-5-6-0-2-8-10-FIN

Camino 5:1-2-3-4-5-7-0-2-8-10-FIN

A modo de ejemplo si se ejecuta el primer camino con los siguientes valores:

Nodo 2:

i = 2, table.getRowCount () = 2

El bucle **for** termina.

Nodo 8:

emptyNets.isEmpty () = true

Nodo 9:

El sistema muestra un mensaje donde anuncia que existen modelos vacíos:

Figura 29 Resultado de ejecutar el primer camino obtenido: elaboración propia

3.3.1.2. Utilización de JUnit para probar el código

Las pruebas de caja blanca catalogadas como dinámicas se pueden realizar usando la herramienta JUnit. A continuación se muestra una imagen con la respuesta de Junit al entrar un grupo de datos mientras se probaban los métodos de la clase CompareController y de aquí se puede deducir que la lógica el programa es correcta cuando los métodos manipulan datos correctos:

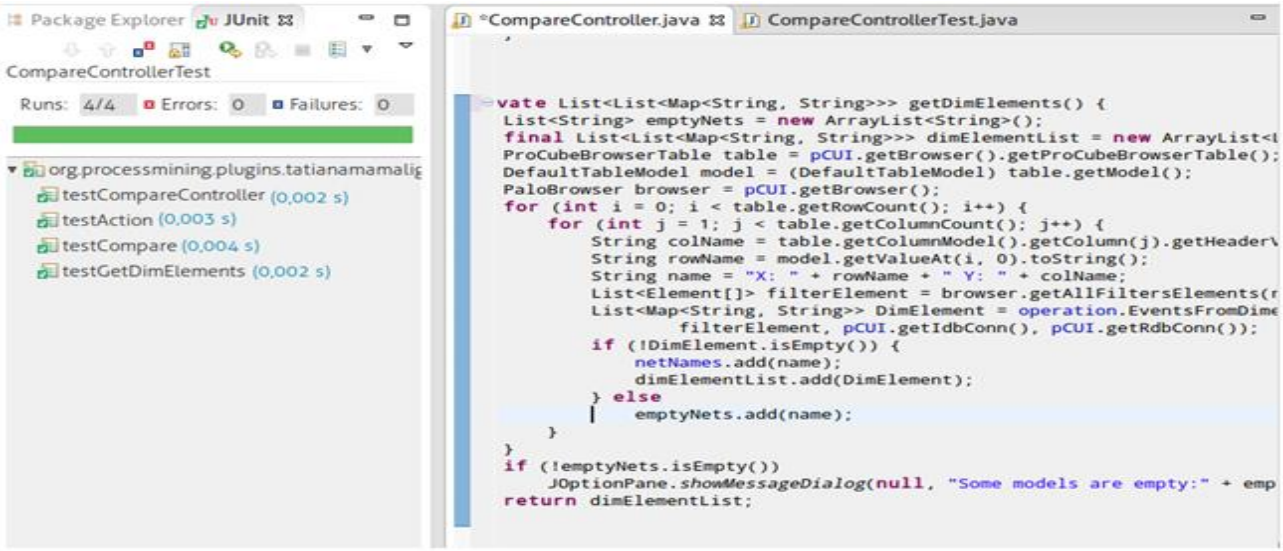

Figura 30 Prueba unitaria con Junit: elaboración propia

3.3.2. Pruebas de Caja Negra

Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software. Este tipo de prueba no es una alternativa a las técnicas de prueba de caja blanca, más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores que los métodos de caja blanca.

La prueba de caja negra encuentra errores de las siguientes categorías (González Varela, 2012):

- Funciones incorrectas o ausentes
- Errores de interfaz
- Errores en estructuras de datos o en accesos a bases de datos externas
- Errores de rendimiento
- Errores de inicialización y de terminación

3.3.2.1. Partición de equivalencia

La partición equivalente es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar (Myers Glenford, y otros, 2011). A continuación se muestra cómo se aplica este método en el presente trabajo de diploma, para el caso de uso Comparar celdas de un Cubo de Procesos:

Escenario	Descripción	Variable 1	Variable 2	Variable 3	Variable 4	Respuesta del sistema	Flujo central
EC 1.1 Válido	Todas las variables son seleccionadas o no se seleccionan algunos atributos ni/o algunas instancias de procesos.	V Seleccionada	V Seleccionada	V Seleccionada	V Seleccionada	El sistema muestra una o varias tablas con la diferencia a nivel de datos entre las instancias de procesos.	Se despliega el menu de Operaciones y se selecciona la opción comparar.
		I No	V Seleccionada	V Seleccionada	V Seleccionada		
		V Seleccionada	I No	V Seleccionada	V Seleccionada		
		V Seleccionada	V Seleccionada	I No	V Seleccionada		
		V Seleccionada	V Seleccionada	V Seleccionada	I No		
		I No	V Seleccionada	I No	V Seleccionada		
		I No	V Seleccionada	V Seleccionada	I No		
		V Seleccionada	I No	I No	V Seleccionada		
		V Seleccionada	I Seleccionada	V Seleccionada	I No		
EC 1.2 Inválido	No se selecciona ningún atributo ni/o ninguna instancia de proceso.	I No	I No	V Seleccionada	V Seleccionada	El sistema muestra un mensaje aclarando que debe seleccionarse al menos una instancia de proceso y un atributo.	
		V Seleccionada	V Seleccionada	I No	I No		
		I No	I No	I No	I No		

Figura 31 Prueba de partición equivalente: elaboración propia

3.4. Nivel de aceptación

La validación del software se consigue mediante una serie de pruebas de caja negra que demuestran la conformidad con los requisitos. El cliente puede validar la solución mediante dos tipos de pruebas, también denominadas pruebas de aceptación (Pressman, 2003):

Prueba Alfa: la prueba alfa se lleva a cabo por un cliente, en el lugar de desarrollo. Se usa el software de forma natural con el desarrollador como observador del usuario y registrando los errores y los problemas de uso. Las pruebas alfa se llevan a cabo en un entorno controlado.

Prueba Beta: la prueba beta se lleva a cabo por los usuarios finales del software en los lugares de trabajo de los clientes. A diferencia de la prueba alfa, el desarrollador no está presente normalmente. El cliente registra todos los problemas que encuentra durante la prueba beta e informa al desarrollador.

En el presente trabajo de diploma se aplicó específicamente la prueba alfa, generando los siguientes artefactos:

Tabla 3 Caso de prueba de aceptación 1

Caso de Prueba de aceptación 1
Caso de uso: Comparar celdas de un Cubo de Procesos
Descripción: la prueba debe corroborar que la aplicación muestre una comparación a nivel de datos de varias celdas.
Condiciones de ejecución: debe existir un registro de eventos que provea los datos para la comparación y se debe haber conformado el Cubo de Procesos.
Entrada/pasos de ejecución: 1. Se despliega el menú "*operations*" 2. Se selecciona la opción "*compare*". 3. Se configuran los parámetros de comparación.
Resultado esperado: se muestran varias tablas con la comparación entre las diferentes celdas y brinda la opción de graficar el resultado.
Evaluación de la Prueba: satisfactoria
Responsable: Omar Sablón Mirabal

Tabla 4 Caso de prueba de aceptación 2

Caso de Prueba de aceptación 2
Caso de uso: Importar registros de eventos
Descripción: la prueba debe corroborar que la aplicación importe un registro de eventos y lo convierta en un cubo.
Condiciones de ejecución: debe existir un registro de eventos.
Entrada/pasos de ejecución: 1. Se despliega el menú "*file*" 2. Se selecciona la opción "*import*". 3. Se busca el registro de eventos y se presiona el botón "aceptar". 4. En la ventana emergente se pone el nombre que tendrá el cubo (almacén de datos). 5. Se seleccionan los atributos que tendrá dicho almacén.
Resultado esperado: se crea un cubo con los atributos deseados y se muestra a la izquierda de la aplicación para poder ser manipulado por el usuario.
Evaluación de la Prueba: satisfactoria
Responsable: Omar Sablón Mirabal

Tabla 5 Caso de prueba de aceptación 3

Caso de Prueba de aceptación 3
Caso de uso: Visualizar los modelos de Celdas
Descripción: la prueba debe corroborar que la aplicación visualice los modelos correspondientes a las celdas.
Condiciones de ejecución: debe existir un cubo para visualizar.
Entrada/pasos de ejecución: 1. Se despliega el menú "*cube*" 2. Se selecciona la opción "*view*". 3. Se selecciona un algoritmo por el cual visualizar.
Resultado esperado: se visualizan los modelos de las celdas y el modelo general del cubo.
Evaluación de la Prueba: satisfactoria
Responsable: Omar Sablón Mirabal

Tabla 6 Caso de prueba de aceptación 4

Caso de Prueba de aceptación 4
Caso de uso: Aplicar operaciones OLAP
Descripción: la prueba debe corroborar que el Complemento aplique las operaciones OLAP.
Condiciones de ejecución: debe existir un cubo al cual aplicarle las operaciones.
Entrada/pasos de ejecución: 1. Se presiona un clic izquierdo sobre uno de los ejes del cubo (x o y) para aplicar las operaciones "*Dice* y *Slice*". 2. Al salir la ventana de confirmación se presiona el botón "ok". 3. En la nueva ventana se escogen los atributos que serán afectados por la operación. 4. Se presiona el botón "*Drill-Down*" o "*Roll-Up*" en esta misma ventana en caso de querer aplicar estas operaciones. 5. Se presiona el botón "ok".
Resultado esperado: se forma un nuevo cubo particionado de la forma deseada por el usuario, con los atributos escogidos por el mismo.
Evaluación de la Prueba: satisfactoria

Responsable: Omar Sablón Mirabal

Se realizó una iteración de prueba y no se detectaron no conformidades por lo que no se realizaron iteraciones adicionales.

3.5. Comparación entre las herramientas convencionales y el CP

En pos de demostrar hasta dónde la solución propuesta resuelve el problema planteado al inicio de la investigación de manera que supere a la mayoría de las herramientas en esta área en cuanto a rapidez y calidad del análisis, se realizó un análisis sobre el registro de eventos del departamento de control de tráfico de Italia, teniendo en cuenta la dimensión tiempo, para detectar las diferencias en cuanto al comportamiento del proceso en períodos diferentes.

Para mejorar el entendimiento se muestra en el anexo1 el modelo de proceso correspondiente al registro de eventos original sin que se le hayan aplicado criterios de comparación, por lo que el modelo muestra el comportamiento del proceso de inicio a fin.

Analizando el modelo siguiente se puede apreciar que en el período de Enero y Febrero de 2007 existen procesos incompletos. O bien se terminan procesos iniciados anterior a esa fecha o se comienzan nuevos procesos sin concluir.

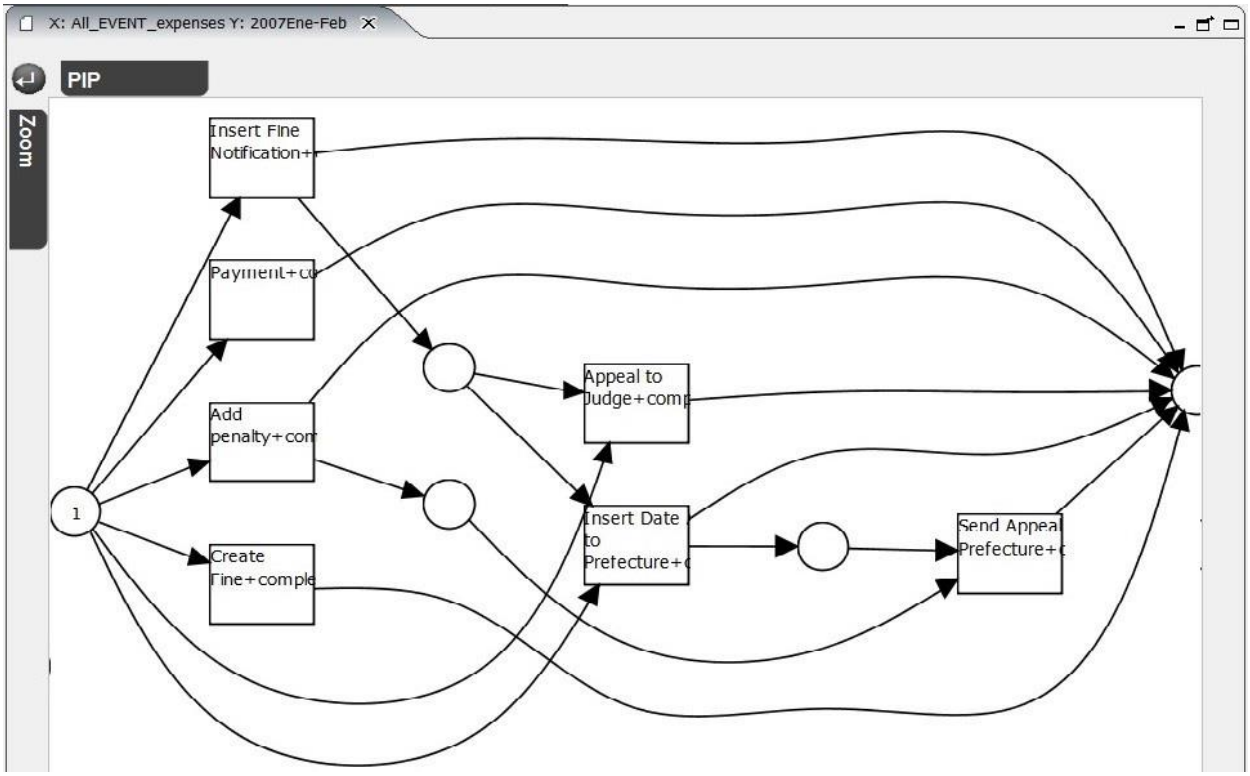

Figura 32 Modelo correspondiente al período Enero-Febrero: elaboración propia

En los datos asociados a ese modelo se aprecia que la actividad “añadir penalización” se realiza con bastante frecuencia y ninguna ocurrencia está relacionada con las multas creadas en ese período. De lo que se concluye que los choferes no suelen pagar las multas creadas en los dos meses anteriores, pues se conoce que un chofer tiene dos meses hábiles para pagar una multa.

Este análisis se muestra en la siguiente tabla donde el período Enero-Febrero está representado por el “Log2” y Julio-Agosto por el “Log4”:

Event Data

Activity	Occurrences		Occurrences %		Resources		Resources %	
	Log2	Log4	Log2	Log4	Log2	Log4	Log2	Log4
Insert Fine Notification	406.0	51.0	37.84	37.78	0.0	0.0	0.0	0.0
Send for Credit Collection	-	-	-	-	-	-	-	-
Payment	197.0	45.0	18.36	33.33	0.0	0.0	0.0	0.0
Insert Date Appeal to Prefecture	40.0	1.0	3.73	0.74	0.0	0.0	0.0	0.0
Send Appeal to Prefecture	10.0	4.0	0.93	2.96	0.0	0.0	0.0	0.0
Appeal to Judge	4.0	1.0	0.37	0.74	1.0	1.0	16.67	100.0
Create Fine	40.0	-	3.73	-	5.0	-	83.33	-
Add penalty	376.0	4.0	35.04	2.96	0.0	0.0	0.0	0.0
Send Fine	-	13.0	-	9.63	-	0.0	-	0.0
Notify Result Appeal to Offender	-	9.0	-	6.67	-	0.0	-	0.0
Receive Result Appeal from Prefecture	-	7.0	-	5.19	-	0.0	-	0.0
Total	1073.0	135.0	-	-	6.0	1.0	-	-

Figura 33 Comparación entre los períodos Enero-Febrero y Julio-Agosto: elaboración propia

Para encontrar una explicación a este suceso se analiza el modelo de Noviembre-Diciembre del 2006, y como se puede apreciar en la siguiente imagen, existen procesos iniciados incompletos, lo que significa que se crean multas que no son pagadas:

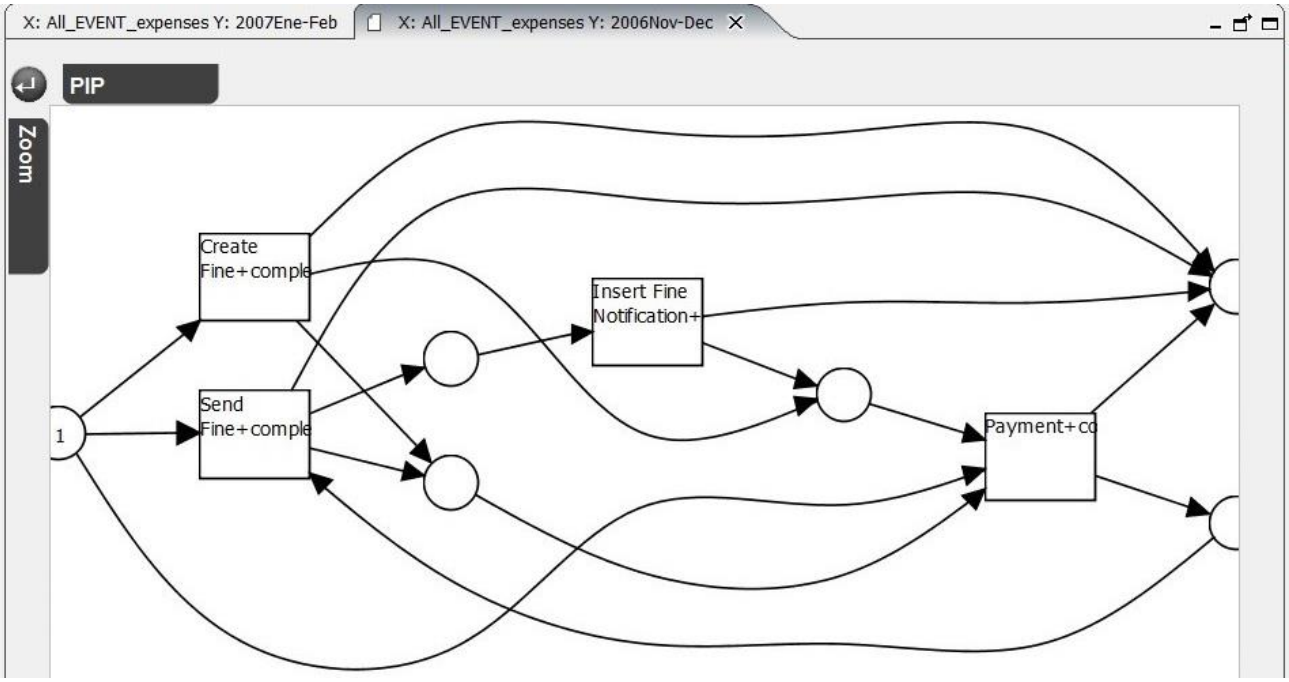

Figura 34 Modelo correspondiente al período Noviembre-Diciembre: elaboración propia

Analizando los datos asociados a este modelo se infiere que el promedio del monto de las multas en noviembre y diciembre es superior a los 22 euros que se consideran aceptables en el

departamento. En esta imagen el período Noviembre-Diciembre está representado por el “Log1” y Enero-Febrero por el “Log2”.

EVENT_amount

Activity	Sum		Sum %		Average		Median		Mode		St. Deviation	
	Log1	Log2	Log1	Log2	Log1	Log2	Log1	Log2	Log1	Log2	Log1	Log2
Insert Fine Notification	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Send for Credit Collection	-	-	-	-	-	-	-	-	-	-	-	-
Payment	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Insert Date Appeal to Prefecture	-	0.0	-	0.0	-	0.0	-	0.0	-	0.0	-	0.0
Send Appeal to Prefecture	-	0.0	-	0.0	-	0.0	-	0.0	-	0.0	-	0.0
Appeal to Judge	-	0.0	-	0.0	-	0.0	-	0.0	-	0.0	-	0.0
Create Fine	38544.0	880.0	100.0	3.33	33.69	22.0	35.0	22.0	35.0	22.0	4.45	0.0
Add penalty	-	25579.0	-	96.67	-	68.03	-	71.5	-	71.5	-	9.43
Send Fine	0.0	-	0.0	-	0.0	-	0.0	-	0.0	-	0.0	-
Notify Result Appeal to Offender	-	-	-	-	-	-	-	-	-	-	-	-
Receive Result Appeal from Prefecture	-	-	-	-	-	-	-	-	-	-	-	-
Total	38544.0	26459.0	-	-	-	-	-	-	-	-	-	-

Figura 35 Comparación entre períodos Noviembre-Diciembre y Enero-Febrero: elaboración propia

Retomando la comparación inicial, pero analizando esta vez el modelo de los meses de Julio y Agosto se aprecia que no se inicia ningún nuevo proceso, solamente se terminan procesos iniciados anteriormente, por ejemplo, aquellos procesos cuyas multas fueron apeladas anteriormente se terminan en este período:

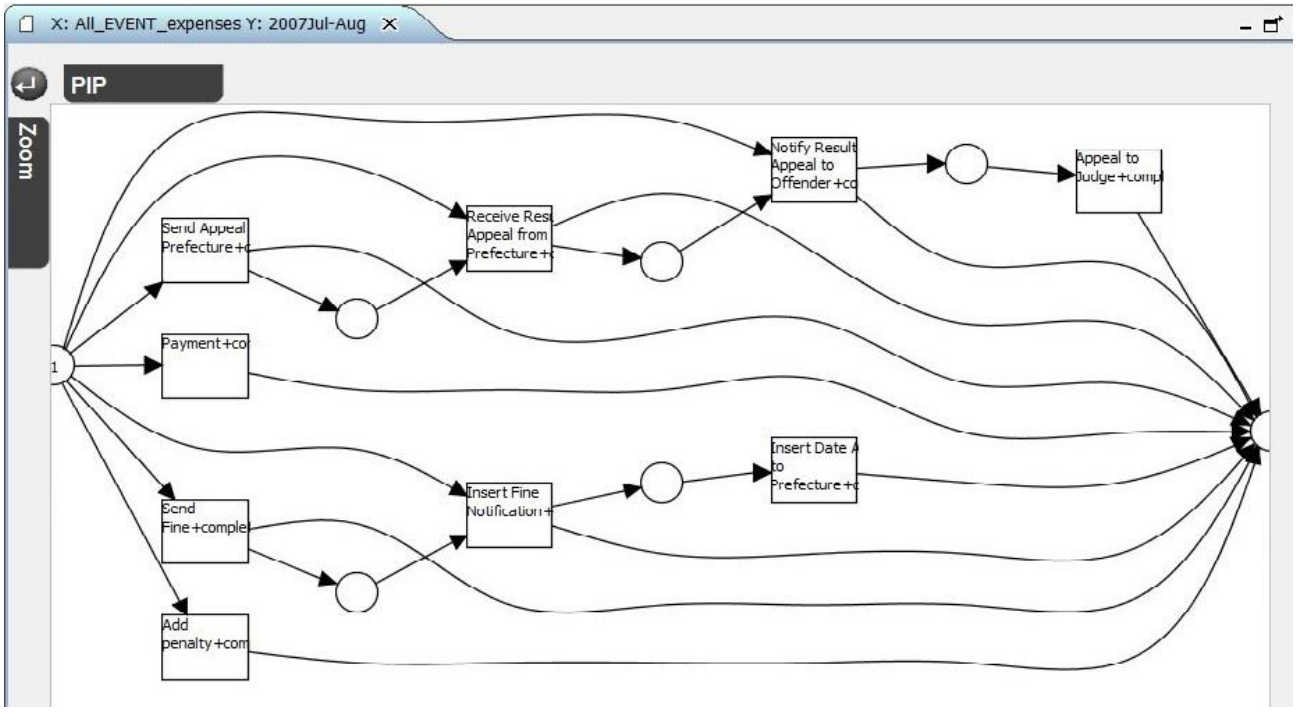

Figura 36 Modelo correspondiente al período Julio-Agosto: elaboración propia

Cuando se analizan los datos se puede ver no solo que en este período se terminan apelaciones, sino que también en estos meses es cuando se responden la mayoría de las apelaciones del año. Además, en estos meses no sólo no se crean multas sino que se añaden muy pocas penalizaciones. En la siguiente imagen el “Log0” representa las ocurrencias totales de las actividades en el año y el “Log3” representa solamente las ocurrencias de Mayo y Junio:

Event Data

Activity	Occurrences			Occurrences %	
	Log0	Log3	Log4	Log0	Log3
Insert Fine Notification	1195.0	4.0	51.0	10.85	0.77
Send for Credit Collection	782.0	-	-	7.1	-
Payment	2109.0	116.0	45.0	19.15	22.39
Insert Date Appeal to Prefecture	79.0	15.0	1.0	0.72	2.9
Send Appeal to Prefecture	143.0	20.0	4.0	1.3	3.86
Appeal to Judge	6.0	-	1.0	0.05	-
Create Fine	3458.0	-	-	31.4	-
Add penalty	1190.0	288.0	4.0	10.81	55.6
Send Fine	2019.0	69.0	13.0	18.34	13.32
Notify Result Appeal to Offender	15.0	1.0	9.0	0.14	0.19
Receive Result Appeal from Prefecture	15.0	5.0	7.0	0.14	0.97
Total	11011.0	518.0	135.0	-	-

Figura 37 Comparación entre el período Julio-Agosto y el año 2007 completo: elaboración propia

Se analizaron además los meses de Mayo y Junio. Al hacer esto se aprecia un comportamiento similar a Julio y Agosto. Lo que demuestra un gran periodo de inactividad en la creación de multas.

Event Data

Activity	Occurrences		Occurrences %		Resources		Resources %	
	Log3	Log4	Log3	Log4	Log3	Log4	Log3	Log4
Insert Fine Notification	4.0	51.0	0.77	37.78	0.0	0.0	0.0	0.0
Send for Credit Collection	-	-	-	-	-	-	-	-
Payment	116.0	45.0	22.39	33.33	0.0	0.0	0.0	0.0
Insert Date Appeal to Prefecture	15.0	1.0	2.9	0.74	0.0	0.0	0.0	0.0
Send Appeal to Prefecture	20.0	4.0	3.86	2.96	0.0	0.0	0.0	0.0
Appeal to Judge	-	1.0	-	0.74	-	1.0	-	100.0
Create Fine	-	-	-	-	-	-	-	-
Add penalty	288.0	4.0	55.6	2.96	0.0	0.0	0.0	0.0
Send Fine	69.0	13.0	13.32	9.63	0.0	0.0	0.0	0.0
Notify Result Appeal to Offender	1.0	9.0	0.19	6.67	0.0	0.0	0.0	0.0
Receive Result Appeal from Prefecture	5.0	7.0	0.97	5.19	0.0	0.0	0.0	0.0
Total	518.0	135.0	-	-	0.0	1.0	-	-

Figura 38 Comparación entre los períodos Mayo-junio y Julio-Agosto: elaboración propia

En conclusión se puede afirmar que los modelos de los períodos Enero-Febrero y Julio-Agosto tienen diferencias estructurales bien marcadas que reflejan comportamientos diferentes del proceso que se evidencian aún más al analizarlo a nivel de datos.

El análisis anterior se llevó a cabo utilizando los Cubos de Procesos. Éste mismo análisis se realizó utilizando Complementos convencionales del marco de trabajo ProM. A continuación se describen los pasos a seguir para realizar el análisis por los Complementos convencionales:

Primero se necesita dividir el registro de eventos según las trazas comprendidas en el período Enero-Febrero. Esto se hace de la siguiente manera:

1. Importar el registro de eventos y seleccionar el Complemento “LTL Checker”.
2. Luego seleccionar las formulas “eventually_after_time_T” y “eventually_before_time_T”.
3. Proveer la fecha 2007/01/01 como el primer parámetro “T” y la fecha 2007/02/28 como el segundo parámetro “T”.
4. Chequear la fórmula.

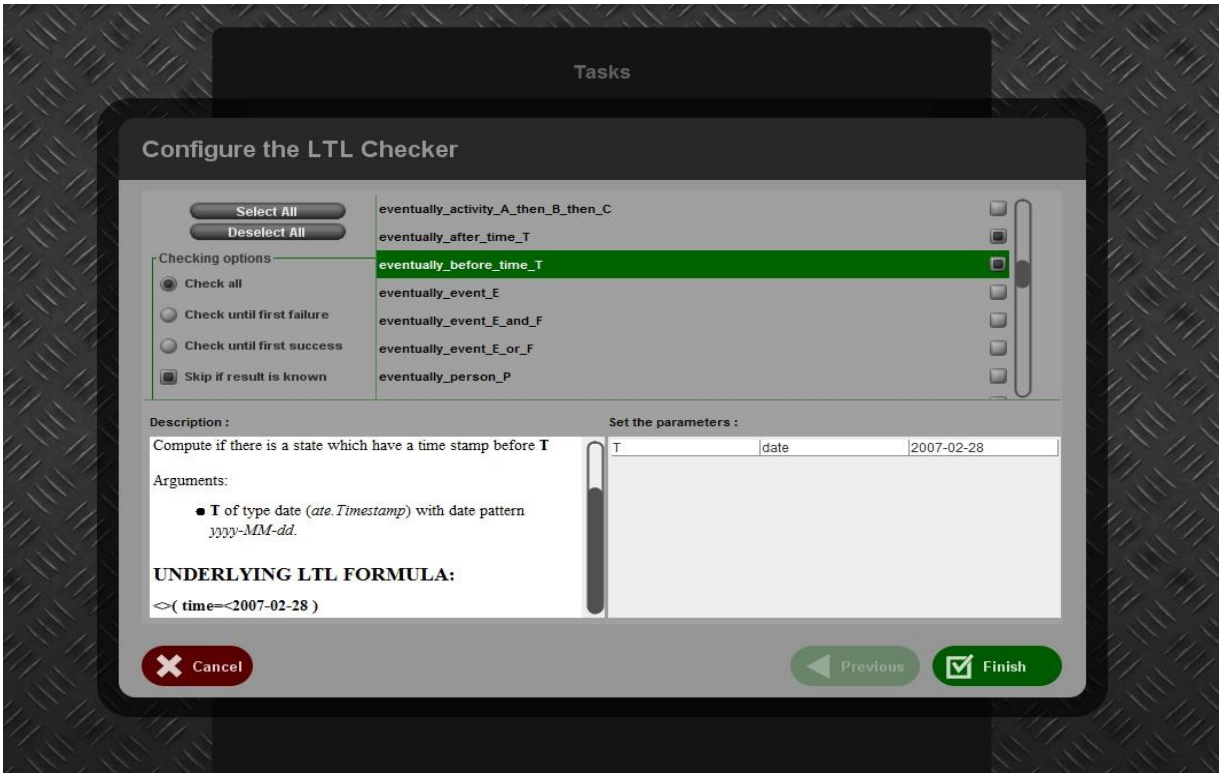

Figura 39 Uso del Complemento LTL Checker: elaboración propia

5. Seleccionar Exportar-> “Instancias correctas” para exportar el registro de eventos en el que estos criterios se satisfacen.
6. Seleccionar Exportar-> “Instancias incorrectas” para exportar el registro de eventos en el que no se satisfacen los criterios.

Después de haber dividido el registro de eventos por estos criterios, se debe visualizar el resultado usando algunos Complementos disponibles para esto, en este caso se usó el “Inductive Visual Miner”. Luego se recolectan algunos datos de forma limitada como las ocurrencias de cada actividad, esto se hace por cada parte resultante de dividir un registro de eventos. El Complemento que realiza esta función es el “Log Visualizer”. Los pasos se describen mejor a continuación:

1. Importar el registro de eventos del paso 5.
2. Modelar el registro usando “Inductive Visual Miner “·

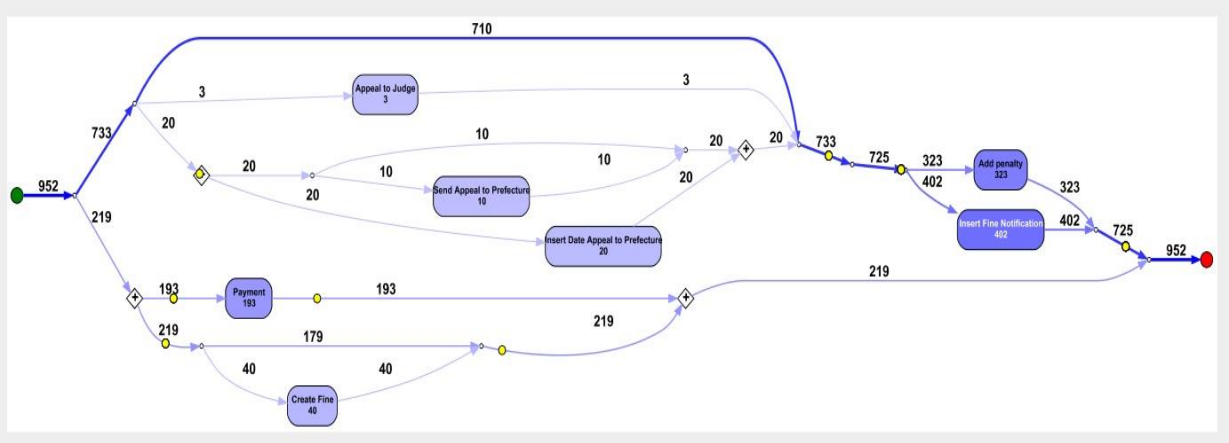

Figura 40 Visualización mediante el Complemento Inductive Visual Miner: elaboración propia

3. Visualizar el registro usando “Log Visualizer”.
4. Importar el registro de eventos del paso 6.
5. Modelar el registro usando “Inductive Visual Miner “·
6. Visualizar el registro usando “Log Visualizer”.

Este proceso habría que realizarlo cada vez que se desee añadir un nuevo criterio de separación y hay que tener en cuenta lo limitado de elegir dicho criterio, ya que debe cumplir con las fórmulas predefinidas del Complemento “LTL Checker” y estas fórmulas en ocasiones no brindan los resultados esperados por la persona que analiza limitando a su vez la capacidad de análisis.

Con los CP el proceso es totalmente diferente. Una vez importado el registro de eventos en la base de datos, este se puede dividir y modelar tantas veces como se desee sin necesidad de importarlo nuevamente. El criterio de separación puede ser con respecto a cualquier atributo contenido en el registro de eventos o con cualquier posible valor de estos, como puede apreciarse esto constituye un amplio diapasón de criterios solo limitado por las características del registro de eventos. Siguiendo el ejemplo anterior:

1. Se seleccionan los atributos del registro de eventos que pasarán a formar las dimensiones del CP, en este caso los atributos “*Amount*” y “*Timestamp*”.
2. Se añade un filtro por cada período que se quiera analizar.
3. Se visualizan los modelos del CP donde se muestran las actividades de los períodos seleccionados.
4. Se visualizan los datos de cada modelo mediante el menú “Compare”.

En general puede apreciarse que el número de pasos necesarios y los Complementos involucrados se acortan considerablemente y como es lógico el tiempo empleado en la preparación de los datos es disminuido. Además la calidad del análisis se aumenta al incluir la posibilidad de realizar un análisis estadístico a los atributos asociados a cada actividad así como visualizar los modelos de forma simultánea para lograr una mejor comparación de los procesos. La gráfica siguiente muestra estos resultados:

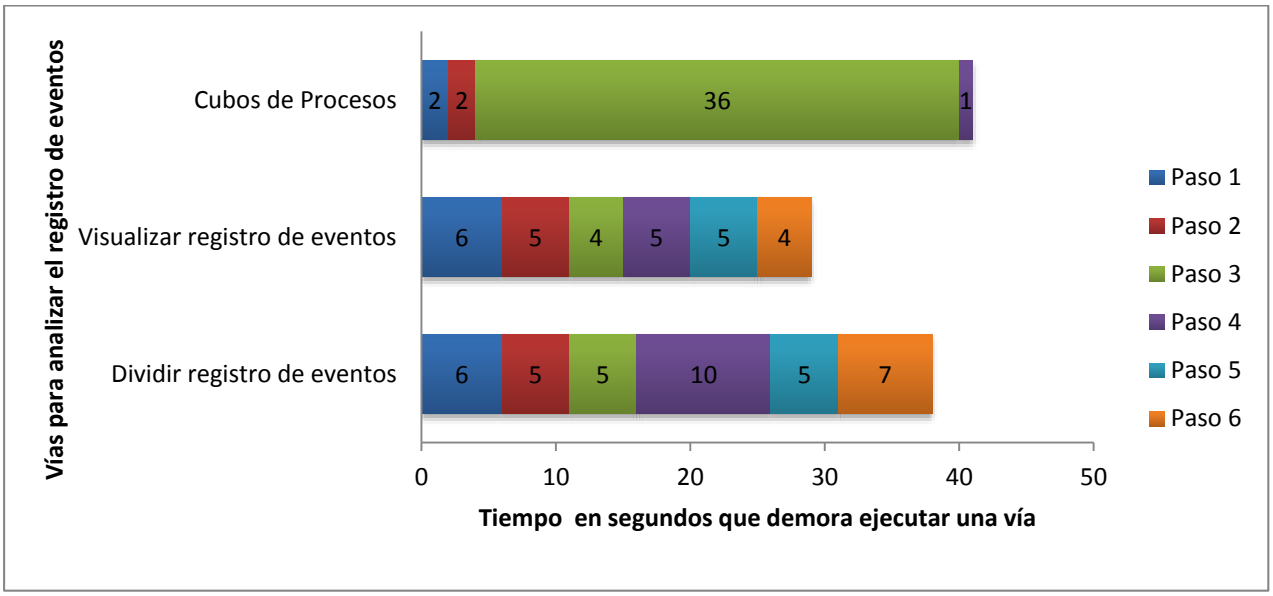

Figura 41 Comparación entre el CP y otras vías para analizar procesos teniendo en cuenta el tiempo que toma ejecutar el análisis: elaboración propia

3.6. Aporte social y económico

La solución creada realiza una comparación en cuanto a los datos de un registro de eventos entre varias celdas. Esto es de vital importancia ya que permite a las empresas tratar con registros de eventos muy grandes que son difíciles de analizar, dividiéndolos en partes más pequeñas. Además, permite personalizar los análisis a los aspectos específicos de un negocio determinado y comparar un proceso grande por partes para descubrir dónde enfocar los esfuerzos. También permite comparar aspectos internos de un negocio buscando eficiencia y eficacia, por ejemplo si se compara el rendimiento de diferentes departamentos se puede descubrir cuál departamento ha trabajado mejor y dónde se consumen más recursos. De la misma forma se pueden encontrar desviaciones en los procesos, anomalías, cuellos de botella, entre otros aspectos relacionados con la seguridad. El hecho de que esta herramienta sea tan completa y que trate aspectos de la Minería de Procesos de manera más eficiente que la mayoría de las herramientas existentes en esta área, implica ganancias para las organizaciones que la utilicen y ventajas competitivas sobre otras que no lo hagan. Todos estos beneficios y muchos más hacen de los Cubos de Procesos una herramienta útil en cualquier ámbito social, desde un hotel hasta un hospital, favoreciendo el diagnóstico de los procesos en la organización.

3.7. Conclusiones parciales

En este capítulo se probó la solución aplicando diferentes técnicas, las de caja blanca destinada al código y las de caja negra a las funcionalidades externas. Cada una de estas técnicas contribuyó a la calidad del producto final y ayudó a encontrar errores invisibles en fases anteriores. Además se validó la solución teniendo en cuenta la opinión del cliente y se corroboró su satisfacción con la solución propuesta. Se realizó una comparación entre las vías convencionales de Minería de Procesos y la solución desarrollada, esta comparación demostró que los Cubos de Procesos son herramientas más eficientes que el resto a la hora de dividir los análisis y de comparar diferentes partes de un proceso.

CONCLUSIONES GENERALES

Al terminar la investigación se puede concluir que:

- La gestión de procesos de negocio, la Minería de Procesos y los Cubos de Procesos favorecen la toma de decisiones, al mismo tiempo que contribuyen a disminuir los problemas en las organizaciones y mejorar la ejecución y resultados de los procesos de negocio.
- Los artefactos generados por AUP facilitaron el desarrollo de una extensión para el Cubo de Procesos que permitiera comparar las celdas a nivel de datos.
- Las pruebas arrojaron que la extensión para el Cubo de Procesos funciona correctamente y que permite realizar comparaciones a nivel de datos entre celdas del cubo, lo que disminuye el tiempo de análisis de registros de eventos muy grandes y brinda evidencias para apoyar la toma de decisiones.

REFERENCIAS

ambysoft. 2015. www.ambysoft.com. [En línea] 2015. http://www.ambysoft.com/.

B. F. van Dongen, A. K. A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and W. M. 2005. *The ProM Framework: A New Era in Process Mining Tool Support.* 2005.

Booch, G., y otros. 2006. *El lenguaje unificado de modelado: guia del usuario.* s.l. : Pearson Addison Wesley, 2006. 9788478290765.

Bose, R.P. Jagadeesh Chandra. 2012. Process Mining in the Large: Preprocessing, Discovery, and. s.l. : Eindhoven University of Technology, 2012.

bpmamericas_org. 2014. bpmamericas_org. [En línea] 2014.

Buijs, J. C. A. M., Dongen, B. F. van y Aalst, W. M. P. van der. 2011. *Towards CrossOrganizational Process Mining in Collections of Process Models and Their Executions.* 2011.

Campodocs. 2012. http://campodocs.com. [En línea] 2012. http://campodocs.com/articulos-informativos/article_69622.html.

Casillas Santillán, Luis Alberto, Gibert Ginestà, Marc y Pérez Mora, Óscar. 2010. *Bases de datos en MySql.* 2010.

Chaudhuri, S., Dayal, U. y Narasayya, V. 2011. *An Overview of Business Intelligence Technology.* 2011.

Club-bpm. 2014. club-bpm.com. [En línea] 2014. http://www.club-bpm.com.

Daniel, Florian, Jianmin, Wang y Weber, Barbara. 2013. *Business Process Management.* Berlin : Springer-Verlag, 2013. 978-3-642-40175-6/0302-9743.

Eclipse.org. 2015. Eclipse.org. [En línea] 2015. http://www.eclipse.org.

Flores, Ervin y Cordero, Jorge Luis. 2014. METODOLOGÍAS ÁGILES “PROCESO UNIFICADO ÁGIL (AUP)". s.l. : Universidad Union Bolivariana, 2014.

Fornaris, Maité Sanchez y Rabí y Alcántara, Dayana. 2010. *Propuesta de una guía de metricas para evaluar el desarrollo de los Sistemas de Información.* 2010.

Frankl, P. G. y Weiss, S. 1994. *An experimental comparison of the effectiveness of branch testing and data flow.* s.l. : IEEE Trans. Software Engineering, 1994.

Fundación universitaria Konrad lorenz. 2014. konradlorenz.edu.co. [En línea] 2014. http://www.konradlorenz.edu.co.

Gonzalez Varela, Luis. 2012. *Introducción al Software Testing.* 2012.

González Varela, Luis. 2012. *Introducción al software testing.* 2012.

Gunter, Christian W. 2009. *Process Mining in Flexible Environments.* Eindhoven : Technische Universiteit Eindhoven, 2009. 978-90-386-1964-4.

Gupta, Monika y Sureka, Ashish. 2015. *Process Cube for Software Defect Resolution.* New Delhi : s.n., 2015.

Hamel, G. y Prahalad, C.K. 1994. *Competing for the future.* Boston : Harvard Business School Press, 1994.

Ibarra, María de los Ángeles. 2005. Procesamiento Analítico en Línea. s.l. : Corrintes, 2005.

Informática-Hoy. 2014. www.informatica-hoy.com.ar. [En línea] 2014. http://www.informatica-hoy.com.ar/telefonos-celulares/Cubo-OLAP-una-base-de-datos-multidimensional.php.

Jagadeesh Chandra Bose, R. y van der Aalst, Will M.P. 2009. *Abstractions in process mining: A taxonomy of patterns.* 2009.

Jagadeesh Chandra Bose, R. y van der Aalst, Will. 2011. *Process Diagnostics Using Trace Alignment:Opportunities, Issues, and Challenges.* 2011.

Janelle, B.H. y Roy Schulte, W. 2011. *BPM Suites Evolve into Intelligent BPM Suites, Gartner.* 2011. G00226553.

JavaWorld. 2012. http://www.javaworld.com/. [En línea] 2012. http://www.javaworld.com/resources.

KDE. 2013. docs.kde.org. [En línea] 2013. https://docs.kde.org/stable/es/kdesdk/umbrello/uml-basics.html.

Larman, Craig. 2010. *UML y patrones.* 2010.

León, E. 2013. Tutorial Visual Paradigm for UML. [En línea] 2013. http://es.scribd.com/doc/36636137/Tutorial-Visual-Paradigm.

Mamaliga, Tatiana. 2013. *Realizing a Process Cube Allowing for the Comparison of Event Data: Master Thesis.* Eindhoven : s.n., 2013.

Manifiesto de Minería de Procesos. **van der Aalst, Wil M.P. 2013.** Eindhoven : s.n., 2013.

Mañas, José A. 2013. Pruebas de Aceptación de Ingeniería de Sistemas Telemáticos. [En línea] 2013. http://www.lab.dit.upm.es/lprg/material/apuntes/pruebas/aceptacion.html.

Mendez, G. 2010. *Ingeniería de Requisitos.* 2010.

Myers Glenford, J., Sandler, Corey y Badgett, Tom. 2011. *The art of the software testing.* s.l. : John Wiley & Sons, 2011.

MySql.com. 2015. MySql.com. [En línea] 2015. http://www.MySql.com.

Patton, R. 2005. *Software testing.* s.l. : Sams Publishing, 2005.

Pérez Alfonso, Damián. 2014. *TÉCNICA PARA EL DIAGNÓSTICO DE VARIANTES DE PROCESOS DE NEGOCIO.* La Habana : Universidad de las Ciencias Informáticas, 2014.

Perez García, Alejandro Alfonso. 2007. *Desarrollo de herramientas web de gestión docente.* s.l. : UNIVERSIDAD POLITÉCNICA DE CARTAGENA, 2007.

Phadke, M. S. 1998. *Planning efficient software tests.* s.l. : Crosstalk, 1998.

Pressman, Roger S. 2003. *Ingeniería de Software: Un enfoque práctico.* 2003.

Process Mining Discovering Workflow Models from Event-Based Data. **Weijters, A.J.M.M. y van der Aslst, Wil M.P. 2001.** 2001.

Sadiq, Shazia, Soffer, Pnina y Völzer, Hagen. 2014. *Business Process Management:12th International Conference, BPM 2014 Haifa, Israel, September 7–11, 2014. Proceedings.* s.l. : Springer, 2014. 978-3-319-10171-2/0302-9743.

Scalaria. 2014. scalaria.co. [En línea] 2014. http://scalaria.co.

Silverio Castro, Rogelio S. 2013. Gestión de Procesos de Negocio. 2013.

Sinexus. 2012. www.Sinexus.com. [En línea] 2012.

Slaughter, Richard A. 2004. *Futures Beyond Dystopia: Creating Social Foresight.* London : RoutledgeFarmer, 2004. 978-0-415-30270-8.

Sommerville, Ian. 2005. *Ingeniería de Software.* Madrid : Pearson Educación, 2005. 84-7829-074-5.

Song, M., Gunter, C. y van der Aalst, Will. 2009. *Trace Clustering in Process Mining.Workshop on Businness Process Intelligence.* Milan : s.n., 2009.

Taniar, David y Chen, Li. 2011. *Integrations of Data Warehousing, Data Mining and Database Technologies.* s.l. : Information Science Reference, 2011.

TU/e. 2015. Process Mining. *Process Mining.* [En línea] 2015. http://www.processmining.org.

Una Perspectiva sobre la Minería de Procesos. **Valle Salas, Antonio y Rozinat, Anne. 2013.** 223, Madrid : ATI, 2013. 0211-2124.

Urquizu, Pau. 2014. www_businessintelligence.com. [En línea] 2014.

Vallecino Moreno, L. y Fuentes Fernandez, L. 2004. *An Introduction to UML Profiles.* 2004.

van der Aalst, Wil M.P. 2013. *Manifiesto de Mineria de Procesos.* Eindhoven : s.n., 2013.

—. 2011. *Process Mining:Discovery, Conformance and Enhancement of Business Processes.* Berlin : Springer, 2011. 978-3-642-19344-6.

van der Aalst, Wil M.P, Pesic, M. y Song, M. 2010. *Beyond Process Mining: From the Past to Present and Future. In Proceedings of the 22nd international conference on Advanced information systems engineering, CAiSE'10.* 2010.

van der Aalst, Wil M.P. 2013. *Process Cubes: Slicing, Dicing, Rolling Up and.* 2013.

van der Aalst, Will M.P. 2012. Mining Process Cubes from Event Data (PROCUBE), project proposal (under review). 2012.

van der Aalst, Will M.P, y otros. 2003. *Data and knowledge Engineering.* 2003.

var der Aalst, Wil M.P. 2014. *Comparative Processes Mining Using Process Cubes.* Eindhoven University of Tecnology, 2014.

Waterman, M. S. 2000. *Introduction to Computational Biology: Maps, Sequence and Genomes.* s.l. : Chapman & Hall/CRC, 2000.

Watson, Andrew. 2013. http://www.uml.org. [En línea] 2013. http://www.uml.org/#Articles.

Weske, Matias. 2007. *Business Process Management: Concepts, Languages, Architectures* . Berlin : Springer-Verlag, 2007. 978-3-540-73521-2.

Yzquierdo Herrera, R., Silverio Castro, R. y Lazo Cortes, M. 2013. *Sub-process discovery: Opportunities for process diagnostics.* s.l. : Springer Berlin Heidelberg, 2013.