Universidad de las Ciencias Informáticas

Facultad 3

Complemento para generar un Árbol de Procesos Configurable

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

Autores: Daniel Pulido Dorta

Carlos Llama Rodríguez

Tutores: Ing. Dina Yaksilik Torres Sakipova

Ing. Damián Pérez Alfonso

La Habana, Cuba

Junio 2014

Pensamiento

Si avanzo sígueme, si me detengo empújame, si retrocedo mátame".

Ché

Declaración de autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.
Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Autores:

_____ _____

Carlos Llama Rodríguez Daniel Pulido Dorta

Tutores:

_____ _____

Ing. Dina Yaksilik Torres Sakipova Ing. Damián Pérez Alfonso

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: Ing. 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

A mis padres por la confianza que siempre tuvieron en mí y por su apoyo incondicional.

A los tutores Dina y Damián por su amistad, guía y apoyo.

Al tribunal por todas sus críticas constructivas y sugerencias.

A mi compañero de tesis por su entrega, sin él este trabajo no hubiese sido lo mismo.

A todos mis compañeros de aula.

A mis amigos de la UCI.

Daniel…

A mi madre por ser mi ángel de la guarda en todos los momentos difíciles de mi vida.

A mi primo Yusek y a Maira por compartir su sabiduría y conocimientos conmigo.

A mis tutores Dina y Damián por estar ahí en todo momento para responder mis infinitas preguntas.

A la profesora Ana Cecilia por atenderme siempre con paciencia y entusiasmo.

A mi compañero de tesis por apoyarme y llegar juntos hasta el final de esta investigación.

A todas las bellas personas de la UCI que hicieron de una forma u otra que este momento llegara.

A mis amigos de toda la vida.

A todos mil gracias.

Carlos…

Dedicatoria

A mi mamá por ser madre y padre a la vez.

A mi esposa Anelys por ser la chica más maravillosa de este mundo y llevar en su vientre a la personita que me hará el padre más feliz del planeta.

A mi familia en general, en especial a mis abuelos Gladys y Oscar.

Carlos…

A mis padres por ser las personas más especiales de mi vida y guiarme siempre por el camino correcto.

A mis abuelos.

A mis tíos.

A mi familia en general.

A mis amigos.

A todos los que de una forma u otra me han apoyado en el transcurso de estos 5 años.

Daniel…

Resumen

La mayoría de las empresas, para la ejecución de sus procesos de negocio, utilizan sistemas de información que generan registros de eventos. Con las técnicas y herramientas desarrolladas en la Minería de Procesos se puede extraer y analizar la información almacenada en ellos, permitiendo diagnosticar un proceso, es decir, analizar el rendimiento, detectar anomalías e identificar patrones de control de flujo en el mismo. Específicamente, la técnica Minería de Variantes descompone un proceso en variantes facilitando su diagnóstico, pero posee como limitante que no visualiza todas las variantes del proceso de forma simultánea. Por ello se desarrolló una técnica para el diagnóstico de procesos, denominada Árbol de Procesos Configurable, que parte de un Árbol de Variantes y lo transforma en un Árbol de Procesos. De esta manera se visualizan todas las variantes del proceso simultáneamente, permitiendo obtener una mejor visión general del mismo. Además, permite configurar sus variantes (bloquear, ocultar o degradar), de manera que se pueden establecer o especificar diferencias en las variantes que permitan diferentes análisis del proceso. Para su desarrollo se utilizó el Entorno de Desarrollo Integrado Eclipse, con el que se conformó el complemento para agregarlo al ProM. Para validar la propuesta se consultaron especialistas que compararon el Árbol de Variantes con el Árbol de Procesos Configurable.

Palabras claves: Árbol de Procesos, Árbol de Procesos Configurable, diagnóstico de proceso, Minería de Proceso, Minería de Variantes, modelos de procesos configurables.

Índice

Índice de figuras

Índice de tablas

Introducción

El desarrollo científico-técnico es uno de los factores más influyentes en la sociedad contemporánea. Según el Dr. Agustín Lage, en la comunidad internacional existe un consenso casi unánime de que a partir de los años 80 del Siglo XX, “la economía de los países desarrollados comenzó a entrar en una etapa diferente, en la que el conocimiento comenzó a ser el activo económico principal. Se le categoriza como una Tercera Revolución Industrial”. Conocida en nuestros días por “Revolución Científico-Tecnológica” (Davila, 2004) .

Esta nueva Revolución en el orden de la Ciencia y la Técnica se ha caracterizado por el surgimiento y la utilización masiva de la computación, la ampliación del uso de la microelectrónica y las telecomunicaciones, la biotecnología, la producción de nuevos materiales y la generación de energías renovables. En esta nueva etapa han tenido una expansión y un impacto extraordinario las nuevas Tecnologías de la Información y las Comunicaciones (TICs).

La economía actual, con una tendencia creciente hacia la globalización, está caracterizada por una creciente competencia entre las empresas por el dominio de los distintos mercados. Por ello, las organizaciones están obligadas a hacer un uso cada vez más eficiente de todos sus recursos, modificando sus estrategias para obtener ventajas competitivas que le permitan diferenciarse dentro de su entorno. Un factor esencial para lograr estos objetivos es la continua innovación de productos y procesos. La adaptación de las empresas a entornos cada vez más complejos, requiere de una inteligencia competitiva y una mejor gestión de la información.

En Cuba se desarrolla un proceso de informatización de la sociedad, impulsado por la actualización del modelo económico, cuya hoja de ruta está marcada por los Lineamientos de la Política Económica y Social del Partido y la Revolución, aprobados como acuerdos del VI Congreso del Partido Comunista de Cuba (PCC), celebrado en abril de 2011.

En el año 2002 se creó la Universidad de las Ciencias Informáticas (UCI), jugando un papel importante en el desarrollo de la Industria Cubana del Software y en la materialización de los proyectos asociados al proceso de informatización de la sociedad. Para ello cuenta con un sistema de enseñanza organizado, proyectos que responden a ese propósito y grupos de investigación creados por estudiantes y profesores, con la meta de convertir a la universidad en una fuente de egreso de personal calificado, comprometido

con la Revolución para apoyar en el proceso de informatización, además de proporcionarle ingresos al país a través de sus proyectos de investigación y desarrollo.

En la Facultad 3, se creó un grupo de investigación sobre la Minería de Procesos (MP), área joven que se encarga de descubrir, monitorear y mejorar los procesos reales que se llevan a cabo en una empresa, extrayendo conocimiento de los registros de eventos que manejan los sistemas informáticos actuales (Aalst, 2012) y generando modelos de procesos. Ha sido aplicada en cientos de organizaciones a nivel mundial en áreas como la banca, telecomunicaciones, salud, comercio y educación, llegando a ser uno de los tópicos de moda en las investigaciones en Gestión de Procesos de Negocio (BPM, del inglés Business Process Management) (Aalst, 2012) y un campo interesante para desarrollar nuevas técnicas para uso profesional, añadiendo valor directo a las empresas. Tanto es así, que en la actualidad existe un enorme interés en la industria por la MP y cada vez más organizaciones aprovechan sus potencialidades (Aalst, 2011).

Según la IEEE Task Force on Process Mining (Fuerza de Trabajo de la IEEE sobre Minería de Procesos) hay dos razones principales para el creciente interés en la Minería de Procesos. Por un lado, se registran cada vez más eventos, proporcionando información detallada acerca de la historia de los procesos. Por otro lado, hay una necesidad de mejorar y apoyar los procesos de negocios en ambientes competitivos y que cambian rápidamente (Aalst, 2011).

Sus técnicas constituyen un medio para chequear y controlar de forma más rigurosa el cumplimiento de normativas, establecer la validez y confiabilidad de la información acerca de los procesos críticos y pocos estructurados de una organización (Aalst, 2012).

En muchas ocasiones los registros de eventos contienen actividades donde varían muchos elementos, como el momento en el que se ejecutaron, el orden de las actividades precedentes y posteriores, y los autores; lo que significa que se generan procesos poco estructurados, que no aportan información debido al alto grado de dificultad que asume comprenderlos.

Existen técnicas desarrolladas hasta el momento que tratan de mitigar las dificultades que presentan los modelos de procesos poco estructurados. Una de estas es la técnica Minería de Variantes (E. P. Hernández, 2014), que da tratamiento a problemas como es el ruido y la incompletitud. Esta técnica genera solamente una variante del proceso a la vez, siendo una variante el camino que va desde el inicio hasta el final del proceso.

Para que el usuario logre observar las demás variantes debe ir desglosándolas por niveles y patrones según su objetivo, constituyendo una vía monótona para el mismo. Debido a que siempre se muestra una sola variante, se dificulta el análisis y entendimiento del proceso en general.

El hecho de no poder contar con todas las variantes del proceso incide negativa y directamente en la detección de actividades como anomalías, fraudes, procesos incompletos, desviaciones, e incluso comprender el proceso. Esta situación no ayuda a distinguir el funcionamiento real de las empresas, encontrar sus errores o dificultades, emprender soluciones y mejoras.

Todos estos problemas repercuten en la meta de la Minería de Procesos que consiste en identificar y extraer rápidamente información importante para apoyar la auditoría y la toma de decisiones para elevar los niveles de competencia del negocio. Por lo tanto, es importante explorar otras vías de solución a la dificultad visual que supone emplear estas técnicas y obtener modelos de procesos.

A partir de esta situación problemática en la presente investigación se identifica el siguiente **problema a resolver:** La generación de un modelo de procesos utilizando la técnica Árbol de Variantes limita la visualización de todas las variantes del proceso de forma simultánea**.**

Objeto de estudio: Minería de Procesos.

Campo de acción: Descubrimiento de Procesos en Sistemas Informáticos.

Objetivo general: Desarrollar un complemento que genere un Árbol de Procesos Configurable a partir de un Árbol de Variantes de un proceso, de manera que visualice todas las variantes del proceso simultáneamente.

Objetivos específicos:

- Realizar la fundamentación teórica de las principales técnicas de visualización de procesos poco estructurados.
- Implementar un complemento que permita obtener un Árbol de Procesos Configurable a partir de un Árbol de Variantes.
- Validar la solución propuesta para comprobar el cumplimiento del objetivo.

Idea a defender:

Si se desarrolla un complemento que genere un Árbol de Procesos Configurable a partir de un Árbol de Variantes de un proceso, se visualizarán todas las variantes del proceso de forma simultánea.

Tareas de Investigación:

1. Estudio de los principales autores de trabajos de mayores resultados en la Minería de Procesos, principales conceptos y temas relacionados.
2. Estudio de los patrones de control de flujo de los procesos.
3. Estudio de los algoritmos existentes empleados en la Minería de Procesos.
4. Estudio de la técnica Árbol de Variantes y Árbol de Procesos Configurable.
5. Diseño de los algoritmos necesarios para transformar los patrones de Árbol de Variantes a Árbol de Procesos Configurable.
6. Implementación de los algoritmos diseñados.
7. Fusión con la herramienta ProM, como marco de trabajo existente para la Minería de Procesos a nivel mundial.
8. Evaluación de la herramienta desarrollada a través de pruebas.

Métodos teóricos empleados:

Histórico-lógico: Este método permite realizar un estudio del estado del arte, comprendiendo la evolución y desarrollo de los temas vinculados a la problemática.

Hipotético-deductivo: Este método permite arribar a conclusiones particulares a partir de formular la hipótesis y después a través de reglas lógicas y deductiva, que posteriormente son sometidas a comprobaciones empíricas.

Modelación: Este método se emplea para contextualizar los elementos involucrados en la visualización de procesos. Es considerado para representar las vistas alternativas de la solución. Mediante este método se crean abstracciones para explicar la realidad de los elementos abordados.

Métodos empíricos empleados:

Medición: Este método es empleado con el objetivo de atribuir valores numéricos a las propiedades y relaciones fijadas por la observación realizada. Se utiliza en las validaciones realizadas en la investigación.

Estructura del trabajo:

Capítulo 1: Fundamentación teórica. En este capítulo se realiza una breve introducción a la Gestión de Procesos de Negocio y se definen los procesos estructurados y poco estructurados. Se exponen conceptos fundamentales de la Minería de Procesos. Además, se describen las herramientas a utilizar en la investigación y se analizan las familias de procesos, árboles de procesos, modelos de procesos configurables, así como los árboles de procesos configurables.

Capítulo 2: Propuesta de solución. En este capítulo se presenta una nueva técnica que a partir de un subproceso descompuesto en variantes construye un modelo en forma de Árbol de Procesos Configurable. Para ello se presentan algunos artefactos generados aplicando la metodología Proceso Ágil Unificado, como el modelo conceptual, las especificaciones de los requisitos funcionales y no funcionales, el diagrama de despliegue, de componentes y de clases. Se valida el diseño realizado aplicando las métricas Tamaño operacional de clases y Relaciones entre clases.

Capítulo 3: Validación de la solución propuesta. En este capítulo se presenta la validación de la solución propuesta. Se aplicó el método consulta a especialistas y se aplicaron pruebas de caja blanca y caja negra al software para comprobar que el mismo cumple con el objetivo trazado en la investigación.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En este capítulo se realiza una breve introducción a BPM. Se exponen los principales conceptos asociados a la Minería de Procesos y sus tipos. Se describen los procesos estructurados y poco estructurados, así como los problemas en el tratamiento y análisis de estos últimos. Se explica en qué consiste el diagnóstico de procesos como parte del ciclo de vida de BPM, las principales técnicas de diagnóstico de procesos existentes y sus principales limitaciones. Además, se describen las familias de procesos, centrándose en la técnica Árbol de Variantes. Se analizan los principales conceptos sobre modelos de procesos configurables, Árbol de Procesos Configurables y Árbol de Procesos. Se describe además el lenguaje y herramientas a utilizar, así como la metodología empleada en la investigación.

1.2 Gestión de Procesos de Negocio

En los estudios realizados en las bibliografías consultadas (Juran, 2000), (Harrington, 2005), (NC/ISO, 2005), (Strnadl, 2005), (Rummler, 2006), (Brache, 2010), se identifican características similares acerca de la definición del concepto de proceso de negocio (Figura 1, tomada de (Strnadl, 2005)), asumiendo como proceso de negocio a un completo y dinámicamente coordinado conjunto de actividades transaccionales y de colaboración que entrega valor a los clientes o se encarga de cumplir otras metas estratégicas de la compañía (Strnadl, 2005) , donde en un contexto de negocios, el propósito de los procesos será proveer a sus clientes internos con sus requerimientos de una manera oportuna (Vázquez, 2007).

Figura 1: Proceso de negocio.

Las entradas son los elementos necesarios que inician el proceso siguiente. Mientras que las salidas son el producto o servicio que los procesos producen (Vázquez, 2007). Las compañías para obtener mejores resultados gestionan sus procesos de negocios.

La Gestión de Procesos de Negocio es una disciplina apoyada fuertemente por tecnologías de información, que combina conocimiento sobre tecnología de información y conocimiento sobre las ciencias de gestión y lo aplica en conjunto a los procesos de negocio (Schunselaar, 2012).

Cuando se diagnostica un proceso, el analista del negocio se encuentra con preguntas interesantes según (Bose, 2010) como:

- ¿Cuál es, probablemente, la conducta del proceso más frecuente que se ejecuta?
- ¿Dónde las instancias del proceso se desvían y qué tienen en común?
- ¿Existen patrones comunes de ejecución en las trazas?
- ¿Cuáles son los contextos en los cuales una o varias actividades son ejecutadas?
- ¿Cuáles son las instancias del procesos que comparten o capturan un comportamiento deseado, ya sea exacto o aproximado?
- ¿Existen patrones específicos (por ejemplo: hitos, actividades concurrentes, etc.) en el proceso?

1.3 Minería de Procesos

Los sistemas de información empleados en la gestión de las organizaciones registran cada vez más los eventos que ocurren durante la ejecución de sus procesos, proporcionando información detallada acerca de la historia de los procesos, lo cual ha despertado un creciente interés en la creación de técnicas y herramientas que permitan estudiar esta información.

Con este surge la Minería de Procesos, disciplina que desarrolla técnicas y herramientas que permiten analizar los registros de eventos, extraer información a partir de ellos y presentar de una forma explícita el conocimiento que contienen (Aalst, 2011).

Un resultado fundamental de la Minería de Procesos es la obtención automática de modelos de procesos. Es importante diferenciar la Minería de Procesos y el modelado estándar que se realiza en las empresas cuando desean describir sus procesos, con ayuda de una herramienta de modelado, como el Visual Paradigm o el Rational Rose por ejemplo, y el conocimiento y la experiencia de una persona calificada. Este modelado presenta las actividades como están establecidas, como deben funcionar y está

determinado por el conocimiento que tenga la persona tanto de los procesos como de la herramienta con la que modela. Sin embargo, la Minería de Procesos presenta las actividades como ocurrieron.

1.3.1 Tipos de Minería de Procesos

La Minería de Procesos permite descubrir los procesos, realizar un análisis de conformidad (investigando si la realidad se ajusta al modelo dado y viceversa) y una extensión (aumentando un modelo existente con información adicional extraída desde los registros de eventos), (Günther, 2009), (Rozinat, 2008), (Yzquierdo-Herrera, 2012), (Schunselaar, 2012), (Bose, 2012).

A partir de los registros de eventos se pueden realizar tres tipos de Minería de Procesos como se muestra en la Figura 2, tomada de (Aalst, 2011).

Figura 2: Tipos de Minería de Procesos.

Tipos de Minería de Procesos: Descubrimiento

Este tipo de Minería de Procesos es el más usado.Toma un registro de eventos y produce un modelo (Günther, 2009) que se descubre sin usar ninguna información a-priori. Existe una gran variedad de investigaciones sobre el descubrimiento.Entre los algoritmos desarrollados se pueden mencionar: Alpha (Medeiros, 2003), (Weijters, 2003), (Aalst, 2004), (Medeiros, 2006), (Günther, 2007), (Goedertier, 2008), (Rozinat, 2008), (Goedertier, 2009), (Aalst, 2010), (WEIJTERS, 2010).

Tipos de Minería de Procesos: Conformidad

La verificación o chequeo de conformidad compara un modelo de procesos existente con un registro de eventos del mismo proceso, permitiendo chequear si la realidad, tal como está almacenada en el registro de eventos, es equivalente al modelo y viceversa. Es decir, permite identificar las desviaciones y anomalías en el proceso ejecutado, pero se requiere de un modelo con el cual comparar el proceso

descubierto y este modelo no siempre existe (Rozinat, 2008), (Günther, 2009), (Dongen, 2010), (Yzquierdo-Herrera, 2012).

Tipos de Minería de Procesos: Mejoramiento

El mejoramiento consiste en extender o mejorar un modelo de proceso existente (Yzquierdo-Herrera, 2012), (Günther, 2009). Su idea se fundamenta en extender o mejorar un modelo de procesos 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 (Aalst, 2011). El diagnóstico ayuda a tener una visión general del proceso, sus aspectos más significativos y de las técnicas que pueden ser más útiles en su posterior análisis (Yzquierdo-Herrera, 2012).

1.4 Tipos de procesos

En la actualidad los procesos tienden a ser poco estructurados. En función de esto, se han creado dos categorías fundamentales de los procesos: proceso lasaña (estructurado) y proceso espaguetis (poco estructurado).

Procesos estructurados

Cuando los registros de eventos muestran que los procesos se caracterizan por una manera de ejecutarse bastante controlada y que poseen una clara estructura, dichos procesos son estructurados y se denominan “procesos lasaña” según (Aalst, 2011), (Bose, 2010), (Bose, 2012), (Song, 2007), (Tiwari, 2008), (Günther, 2007), (WEIJTERS, 2010), (Roya ZarehFarkhady, 2012), (Yzquierdo-Herrera, 2012), (Schunselaar, 2012). Estos procesos son frecuentemente menos interesantes debido a que es fácil aplicarle técnicas de Minería de Procesos y, debido a que están organizados, las mejoras son pequeñas.

Procesos poco estructurados

En muchas ocasiones los registros de eventos contienen actividades ejecutadas con diversos órdenes, donde varía mucho el momento en el que se ejecutaron, las actividades precedentes y posteriores cambian, cambian los autores, o se evidencian muchas relaciones o variaciones entre las actividades. Estos procesos poco estructurados como se muestra en la Figura 3, tomada de (Aalst, 2011), denominados “espagueti” según (Aalst, 2011), solo permiten aplicársele un subconjunto de las técnicas

de Minería de Procesos. Generan modelos poco comprensibles que no aportan información debido al alto grado de dificultad que asume comprenderlos (Günther, 2009), (Bose, 2010), (Bose, 2012), (Song, 2007), (Tiwari, 2008), (Günther, 2007), (WEIJTERS, 2010), (Roya ZarehFarkhady, 2012), (Yzquierdo-Herrera, 2012), (Schunselaar, 2012).

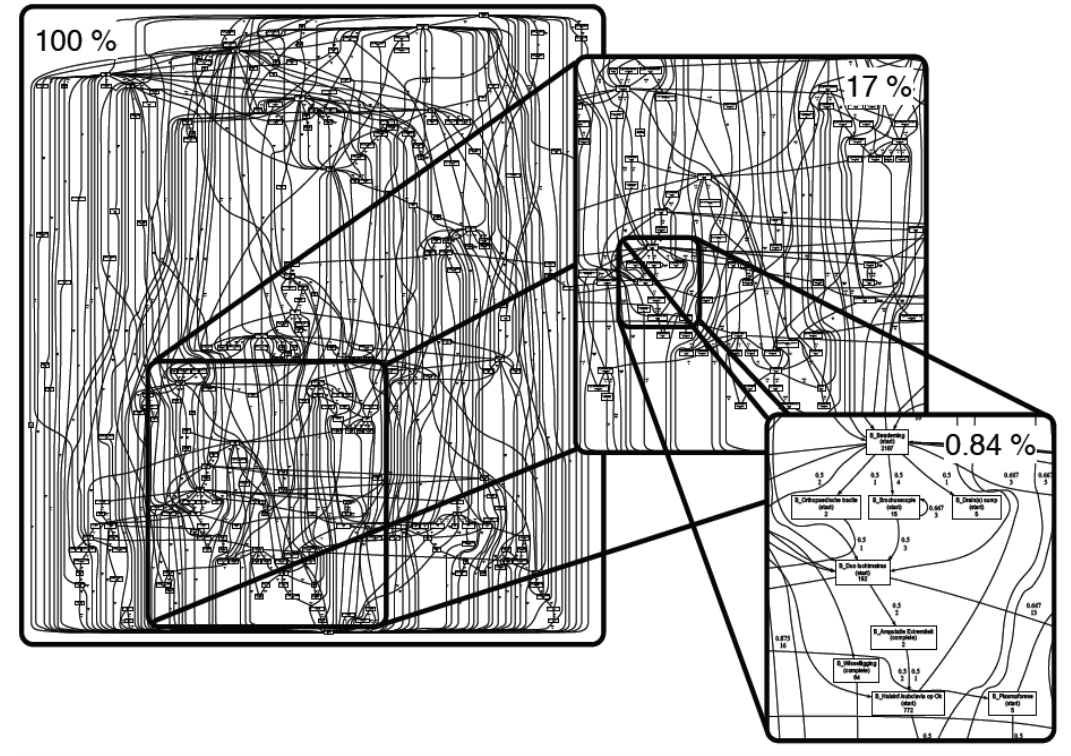


Figura 3: Procesos poco estructurados.

Actualmente no todos los negocios poseen sus procesos bien estructurados. De hecho, los procesos tienden a ser menos estructurados de lo que se espera según (Bose, 2012). Esto influye en el modelo de los procesos de negocio como artefacto de salida del descubrimiento como tipo de Minería de Procesos. Si se trata de construir un modelo para casos muy diferentes, entonces el modelo es probablemente demasiado complicado. Las técnicas de Minería de Procesos que existen, son incapaces de lidiar con esta situación. Por tanto, son creados modelos muy generalizados, donde se incluyen comportamientos no deseados en el proceso (Medeiros, 2006).

Como se mencionó anteriormente, no todos los procesos tienen el mismo nivel de complejidad para ser analizados, lo cual es relevante en el tratamiento y análisis de los procesos.

1.4.1 Problemas en el tratamiento y análisis de los procesos poco estructurados

Algunos de los problemas encontrados (EMIT, 2004), (Tiwari, 2008), (Günther, 2009), (Roya ZarehFarkhady, 2012), (Aalst, 2011), (Aalst, 2012) en la Minería de Procesos son:

- Ruido: Los datos registrados pueden ser incorrectos o incompletos, creando problemas cuando los datos son minados. El registro de eventos contiene comportamiento raro o infrecuente que no representa el típico comportamiento de los procesos.
- Tareas escondidas: Tareas que existen pero que no se pueden encontrar en los datos.
- Incompeltitud: El registro de eventos contiene pocos eventos, o faltantes, como para descubrir estructuras de control de flujo.
- Tareas duplicadas: Dos nodos del proceso pueden referirse al mismo modelo de procesos.
- Constructores sin libre elección: Opciones controladas que dependen de selecciones hechas en otra parte del modelo de proceso.
- Ciclos minados: Un proceso puede ejecutarse varias veces, los ciclos pueden estar simplemente involucrando uno o varios eventos o pueden ser más complejos.
- Distintas perspectivas: Los eventos del proceso pueden añadirse con información adicional con propósitos específicos.
- Visualización de resultados: Pueden presentarse de formas gráficas en paneles de gestión.
- Resultados heterogéneos: Acceso a sistemas de información basados en diferentes plataformas.
- Procesos concurrentes: Procesos que ocurren a la misma vez.
- Búsquedas locales y/o globales: Las estrategias locales restringen el espacio de búsqueda y son menos complejas. Las estrategias globales son complicadas pero tienen mejor chance de encontrar una solución óptima.

Se ha encontrado que muchos de los problemas que enfrentan los enfoques tradicionales de Minería de Procesos pueden estar directamente relacionados con el hecho de que se esfuerzan por la precisión en la descripción del proceso observado. Por intentar describir las observaciones en un modelo, es decir, la lucha por la precisión de la conducta, un gran número de arcos se introducen en el modelo de proceso resultante. Además, teniendo en cuenta todos los tipos observados de la actividad, es decir, la lucha por

la precisión de su alcance, estos algoritmos son propensos a la creación de modelos de procesos grandes y complejos. Se sostiene que, por comprometer la precisión del comportamiento y alcance, los algoritmos de Minería de Procesos pueden producir resultados que son más útiles en el contexto de procesos flexibles.

Se han desarrollado diversas técnicas en la Minería de Procesos que pueden ser de cualquiera de sus tres tipos (descubrimiento, verificación o mejoramiento). Estas técnicas se pueden utilizar en cualquiera de las siete etapas de la gestión de procesos de negocio (Figura 4, tomada de (Aalst, 2011)) con el objetivo de mejorar la comprensión de los procesos pocos estructurados y evitar algunos de los problemas mencionados anteriormente en el *epígrafe 1.4.1 Problemas en el tratamiento y análisis de los procesos poco estructurados*.

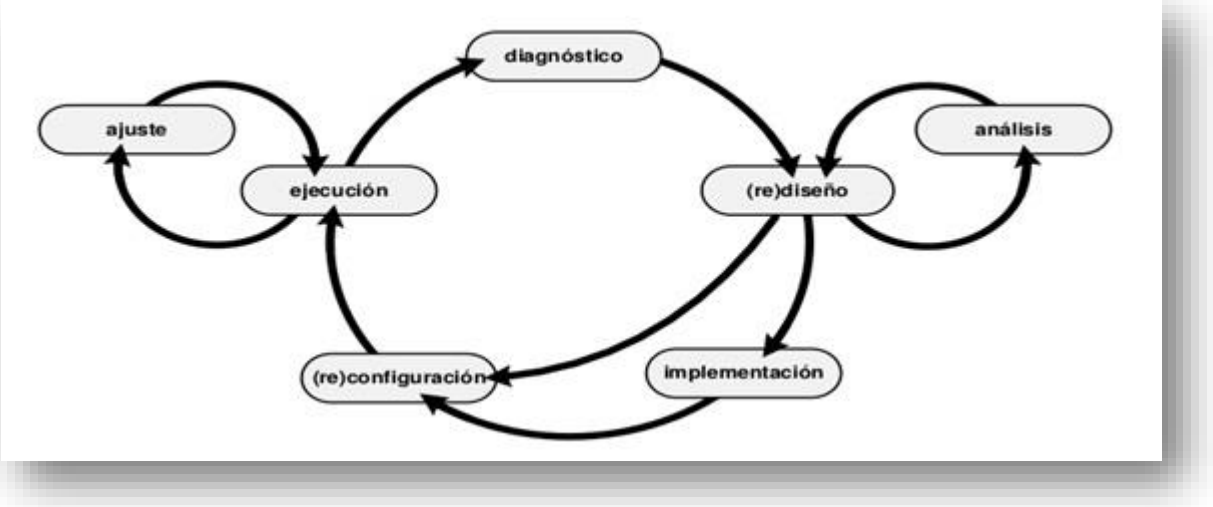


Figura 4: Etapas de la gestión de procesos de negocio.

1.5 Diagnóstico de procesos

Una de las etapas dentro de BPM es el diagnóstico de procesos. El mismo abarca el análisis de rendimiento, la detección de anomalías y la identificación de patrones comunes (BOSE R.P.J.C, 2012). Utilizando técnicas de diagnóstico de procesos en los registros de eventos es posible conocer los comportamientos presentes de mayor frecuencia, lo que posibilita dirigir las técnicas de mejora hacia los elementos más críticos del proceso. Al mismo tiempo, la detección de ejecuciones anómalas existentes

en el registro de eventos brinda información acerca de posibles violaciones del proceso. La presencia de desviaciones en el modelo constituye un indicador de posibles fraudes o violaciones en las políticas y suele ser útil a las organizaciones en la realización de auditorías (E. P. Hernández, 2014).

1.5.1 Técnicas de diagnóstico de procesos

Como parte de la investigación que se llevó a cabo, se analizaron diferentes técnicas de diagnóstico de procesos. Entre ellas se encuentran Análisis de Gráficos de Puntos (Song, 2007)], Minería Difusa (Günther, 2007), Alineación de trazas (Bose, 2012) y Descomposición en subprocesos utilizando bloques de construcción (Yzquierdo-Herrera, 2012). Entre las principales limitaciones identificadas en las técnicas analizadas se pueden mencionar las siguientes:

Análisis de gráficos de puntos

Esta técnica (Figura 5, tomada de (Bose, 2010)) tiene como desventaja la forma en que se grafican los puntos o eventos porque no se encuentran alineados y por lo tanto, es difícil determinar patrones de control de flujo. Esta propuesta presenta una vista “helicóptero” del registro de eventos. Esto no es factible ya que el analista del negocio necesita buscar manualmente en el mapa de puntos para identificar cualquier información potencial del proceso (Bose, 2010). Esto se torna engorroso cuando los registros superan los cientos de actividades y muchas veces no es posible identificar patrones interesantes.

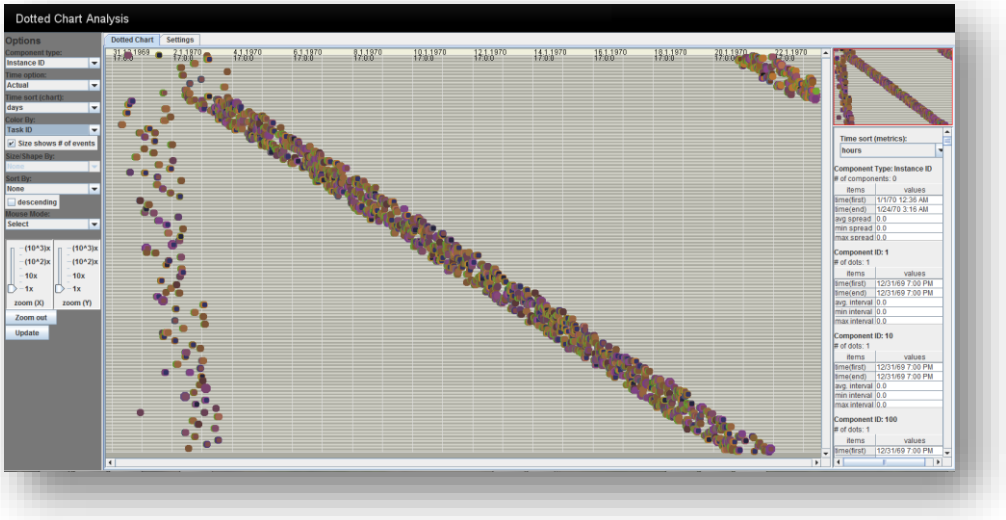


Figura 5: Análisis de gráficos de puntos.

Alineación de trazas (Bose, 2010)

Esta técnica (Figura 6, tomada de (Bose, 2010)) muestra una vista del registro de eventos donde se pueden identificar las desviaciones del proceso pero no enmarca las anomalías y desviaciones identificadas en subprocesos, lo que dificulta su contextualización. También, cuando hay ruido en las trazas el resultado puede ser de baja calidad. Por consiguiente, se necesitan técnicas que identifiquen y desechen el ruido durante la alineación.

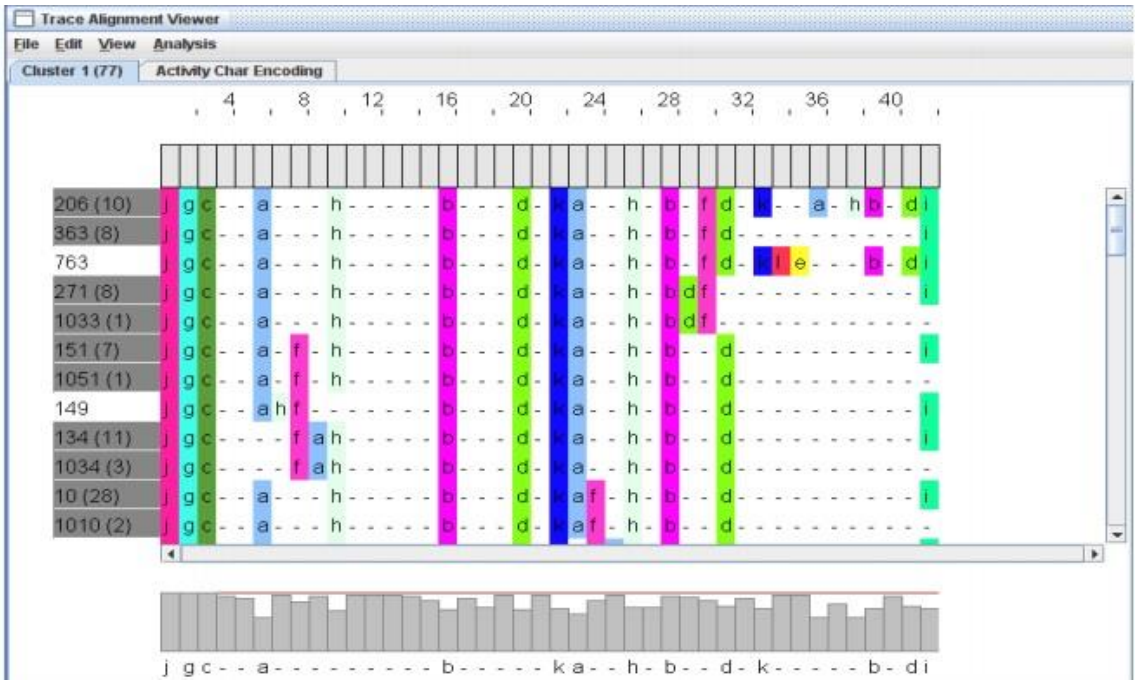


Figura 6: Alineación de trazas.

Descomposición en subprocesos utilizando bloques de construcción (Yzquierdo-Herrera, 2012)

La Descomposición en subprocesos (Figura 7, tomada de (Yzquierdo-Herrera, 2012)) se basa en la técnica de Alineación de trazas, explicada anteriormente. Una limitante de esta técnica resulta la imposibilidad de lidiar con características presentes en los registros de eventos como ruido o ausencia de información y ante registros de eventos muy grandes y muy complejos, y estas generalmente son características que están presentes en los modelos de procesos pocos estructurados.

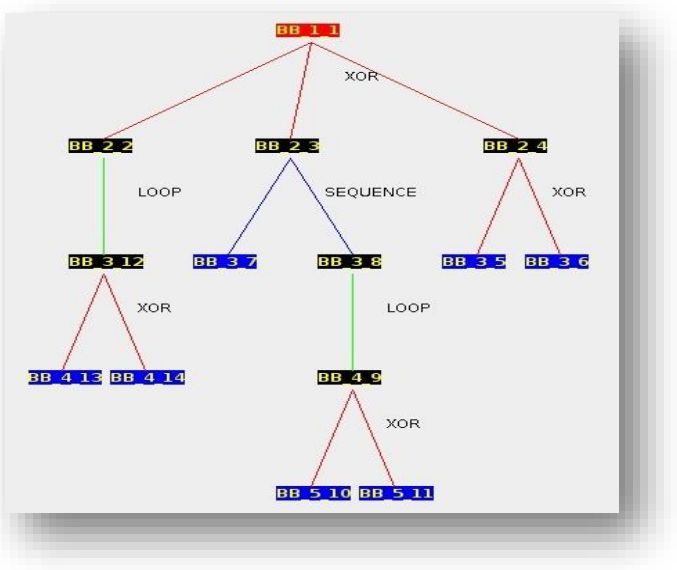


Figura 7: Descomposición en subprocesos.

Modelo difuso

Esta es una técnica para visualizar procesos complejos utilizando la idea de los mapas cartográficos (Figura 8, tomada de (Günther, 2009)), los cuales son bastante imprecisos, es decir, son capaces de abstraerse de los detalles y de agregar comportamiento que no resulta interesante. No pueden ser usados para controlar la ejecución de los procesos debido a que no definen la lógica del mismo de una manera lo suficientemente precisa. Esto es intencional, ya que su objetivo es proveer una simple visualización del proceso, por tanto sacrifica precisión por comprensión y simplicidad. Además, esta técnica requiere de un gran número de configuraciones y varios parámetros resultan confusos para muchos usuarios (Günther, 2009).

Esta técnica se acerca mucho al concepto de familia de procesos, que se explica en el siguiente epígrafe, ya que al abstraerse de detalles está generando una variante del proceso lo que resulta muy difícil de comprender debido a que no se sabe si las opciones desechadas fueron debido a un operador Paralelismo, XOR o a un AND.

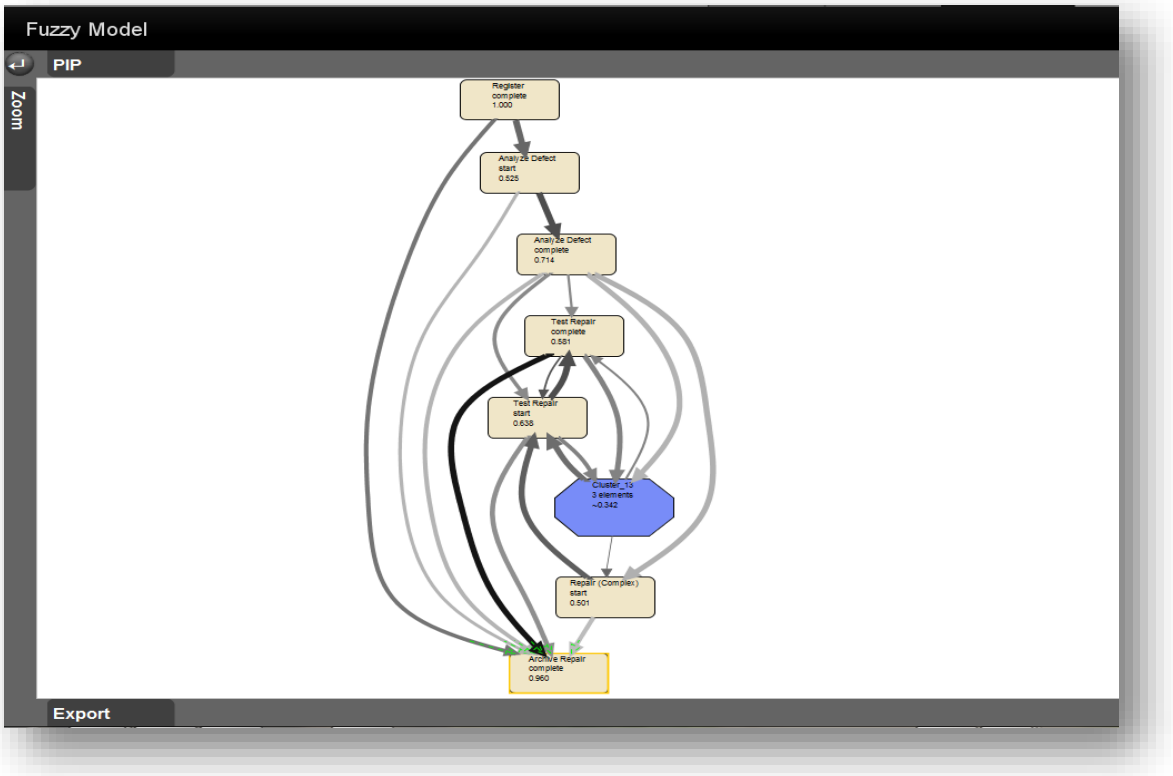


Figura 8: Modelo difuso.

Pero en general todavía los modelos obtenidos resultan difíciles de interpretar. Las técnicas desarrolladas en esta área presentan problemas al identificar los patrones más comunes de ejecución y las desviaciones en el registro de eventos. Una forma de entender un proceso poco estructurado es convertirlo en una familia de procesos mediante una separación de variantes.

1.6 Familia de Procesos

Las familias de procesos son procesos de negocios que se pueden gestionar como familias de variantes (Figura 9, tomada de (La Rosa M., 2013)), permitiendo que los analistas se enfrenten a la elección entre el modelado de cada variante por separado, o modelar múltiples o todas las variantes de un único modelo. El modelado de cada variante por separado conduce a una proliferación de modelos que comparten partes

comunes, dando lugar a redundancias e inconsistencias. Mientras tanto, el modelado de todas las variantes juntas conduce a una menor redundancia e inconsistencia.

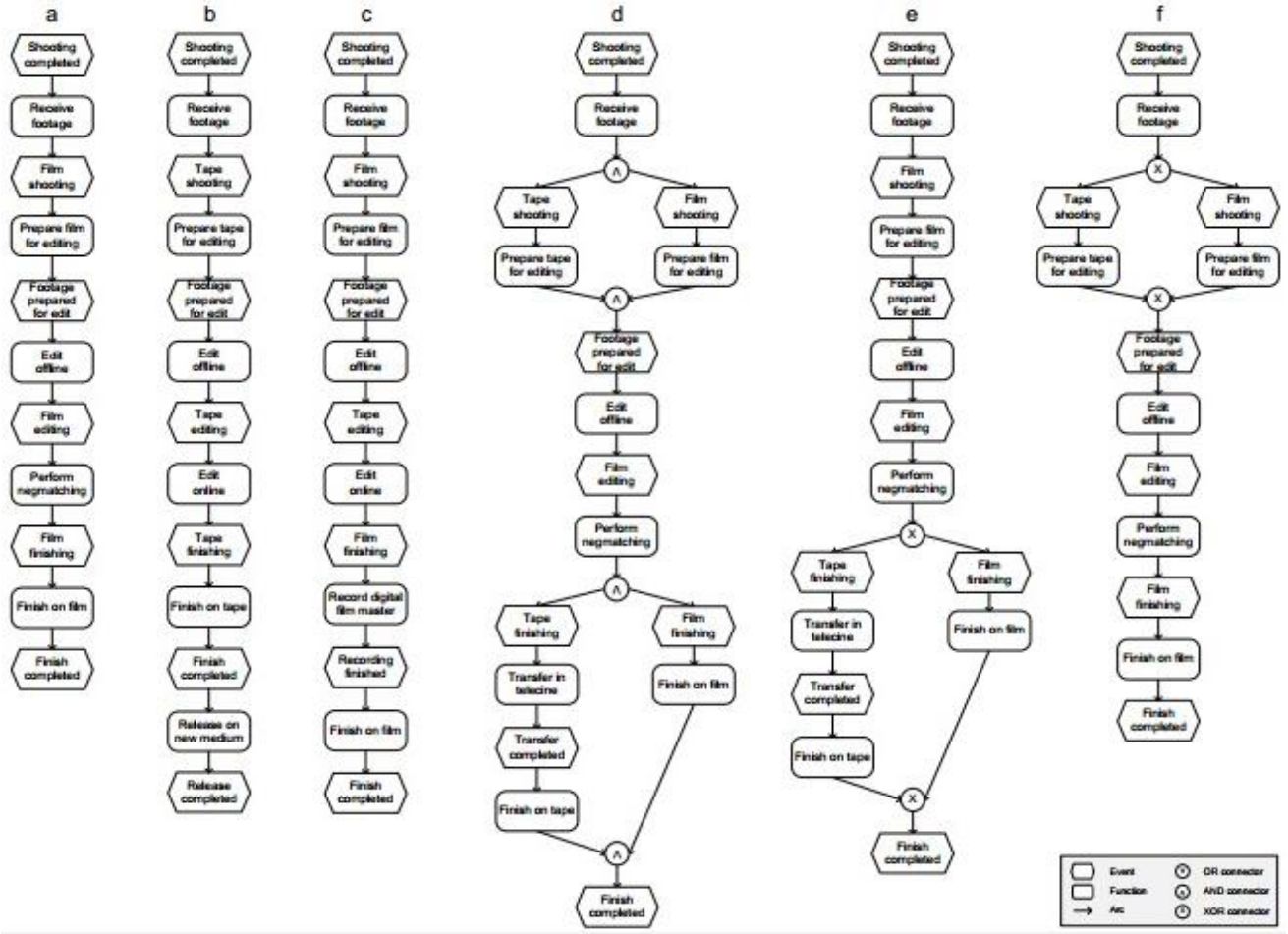


Figura 9: Familia de procesos.

Una forma de descubrir en la etapa de diagnóstico una familia de procesos es mediante la técnica de diagnóstico Minería de Variantes.

1.6.1 Minería de Variantes

Actualmente se encuentra en desarrollo la técnica de diagnóstico de procesos Minería de Variantes (Figura 10, tomada de (E. P. Hernández, 2014)), la cual es una representación de diferentes descomposiciones en subprocesos aplicadas a un proceso P a partir de un registro de eventos. Este árbol está compuesto

por dos tipos de nodos, los nodos subproceso y los nodos patrón. Un nodo subproceso representa un subproceso sobre P y puede o no contener nodos patrones como hijos. Un nodo patrón representa una descomposición de su padre, de acuerdo a un patrón de control de flujo, por lo que un nodo patrón posee dos o más nodos subproceso como hijos. El nodo raíz es un nodo subproceso y está referido a todo el proceso. Los nodos hojas son siempre nodos de tipo subproceso (E. P. Hernández, 2014).

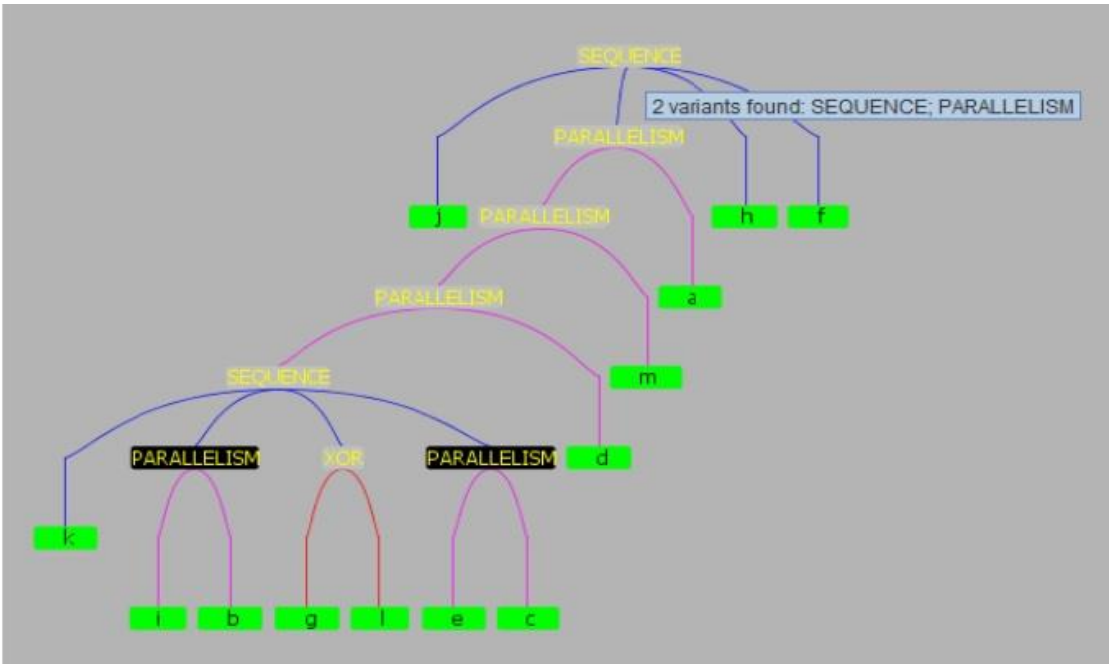


Figura 10: Árbol de Variantes.

Un proceso se puede descomponer en múltiples subprocesos utilizando los siguientes patrones de control de flujo (E. P. Hernández, 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 repiten múltiples veces. Cada repetición comienza con la ejecución del primer subproceso (Do), continúa con el segundo (Redo) y termina con el Do. El Redo puede ser un subproceso vacío, por lo que el único repetido sería el Do.

A continuación se explica otra manera en las que se pueden visualizar las familias de procesos.

1.7 Modelos de procesos configurables

Un modelo de proceso configurable describe una familia de modelos de proceso, es decir, las variantes del mismo proceso. Una configuración de un modelo de proceso configurable restringe su comportamiento, por ejemplo, ocultar o bloquear las actividades, donde ocultar significa que una actividad puede ser omitida y bloquear significa que un camino no se puede tomar más. La mayoría de los formalismos permiten a los operadores ser más restrictivos (por ejemplo, un OR se transforma en un XOR). Mediante la configuración del modelo de proceso configurable se obtiene un modelo de proceso (regular). Un modelo de proceso configurable pretende mostrar similitudes y diferencias entre las distintas variantes. Esto facilita la reutilización y la comparación (Buijs, 2013).

Existen diferentes notaciones y enfoques para representar los modelos de procesos configurables. Los modelos de procesos configurables se pueden construir de diferentes maneras, pueden ser diseñados desde cero, o si una colección de modelos ya existe, entonces el modelo de proceso configurable se pueden obtener mediante la fusión de las diferentes variantes. Existen diferentes enfoques para combinar una colección de modelos de procesos existentes en una modelo de proceso configurable (Buijs, 2013).

En (Aalst, 2011) plantean que las Redes de Petri (Figura 11, tomada de (Aalst, 2011)) constituyen el lenguaje de modelado de proceso más antigua y mejor investigado que permite el modelado de concurrencia. Aunque la notación gráfica es intuitiva y sencilla, las Redes de Petri son ejecutables y muchas técnicas de análisis se pueden utilizar para analizarlos.

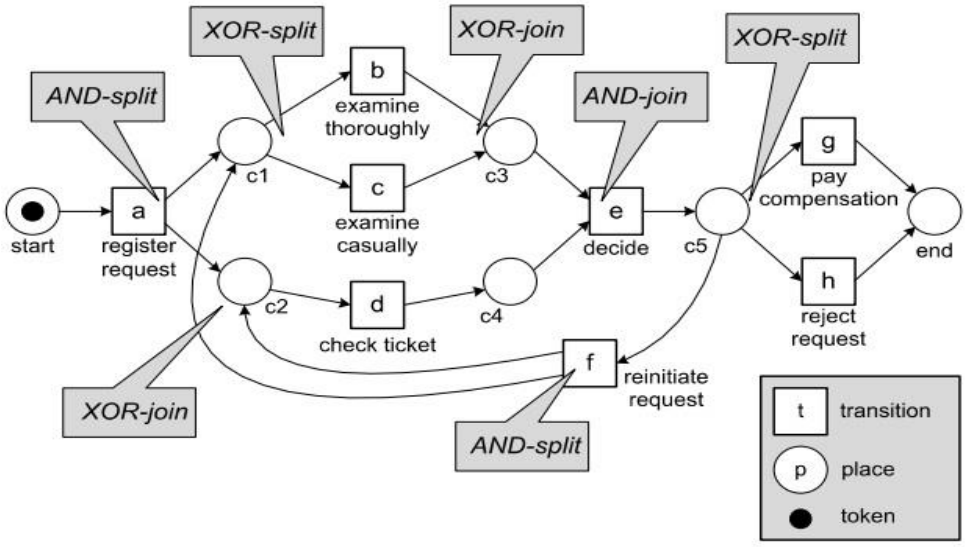


Figura 11: Modelo de proceso utilizando la notación Redes de Petri.

También en (Aalst, 2011) se plantea que la Notación para la Gestión de Procesos de Negocio (BPMN) se ha convertido en uno de las más utilizadas para modelar los procesos de negocio (Figura 12, tomada de (BPM, 2014)).

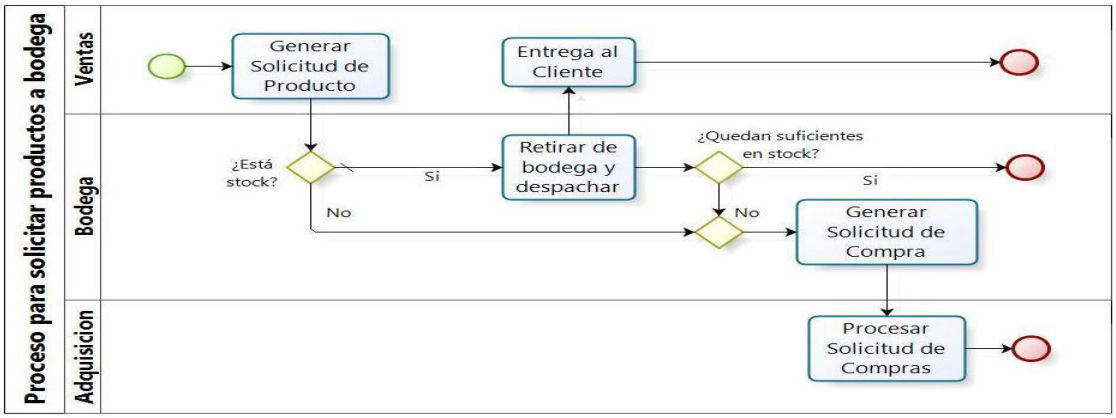


Figura 12: Modelo de proceso utilizando BPMN.

En (Becker, 2004) se centran en cómo la notación de modelado de procesos de negocio Cadenas de Procesos representadas por eventos (EPCs) se puede hacer configurable usando la técnica presentada en (Gottschalk, 2008) donde se propone combinar una colección de EPCs, obteniéndose como resultado

un EPC configurable (Figura 13, tomada de (La Rosa M., 2013)) que permite un comportamiento adicional, no posible en los EPCs originales.

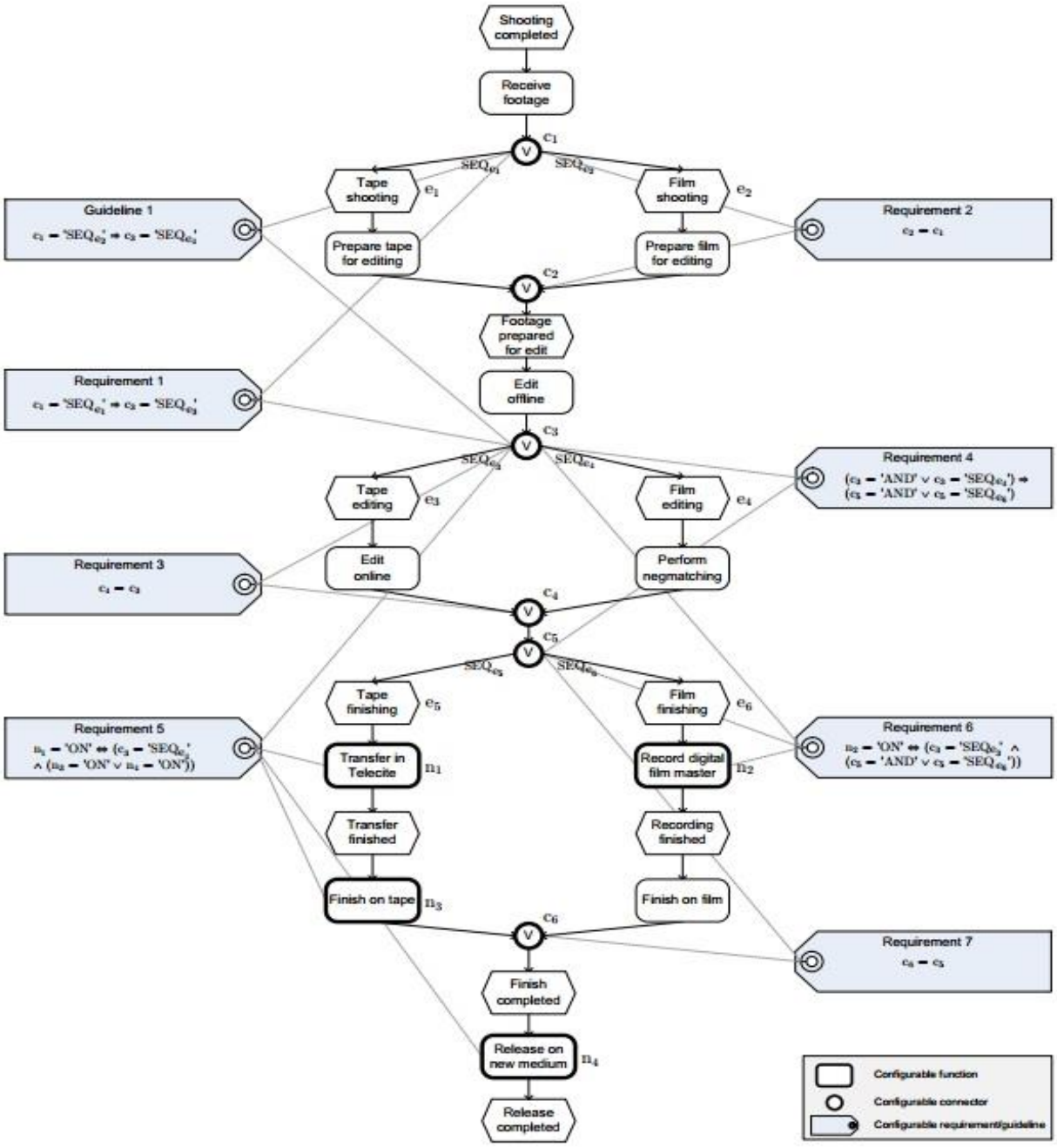


Figura 13: Modelo C-EPC.

En (Gottschalk, 2008) se presenta un enfoque para identificar los elementos configurables de un lenguaje de modelado de flujo de trabajo (YAWL) y oportunidades para agregar configuración a los modelos de flujo de trabajo, donde los elementos configurables son los elementos de un modelo de flujo de trabajo que se pueden modificar de tal manera que el comportamiento representado por el modelo está restringido, es decir, un elemento configurable se puede establecer como activado, como bloqueado , o como oculto, obteniéndose de esta manera un YAWL configurable (Figura 14, tomada de (Gottschalk, 2008)).

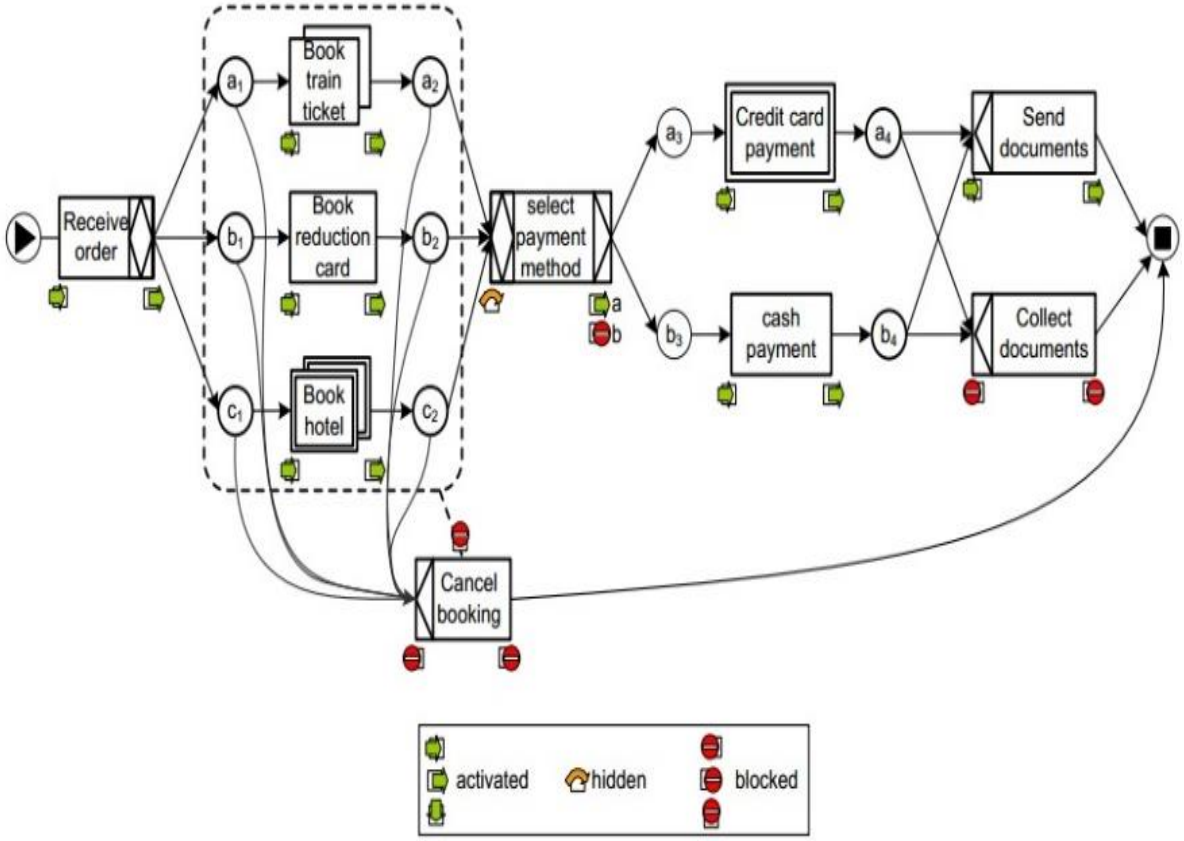


Figura 14: Representación de un YAWL configurable.

En (Schunselaar, 2012) se plantea el enfoque CoSeNet, el cual ha sido diseñado para combinar una colección de modelos de procesos estructurados de bloque. Este enfoque siempre resulta en modelos de procesos configurables acertados y reversibles (Figura 15, tomada de (Schunselaar, 2012)).

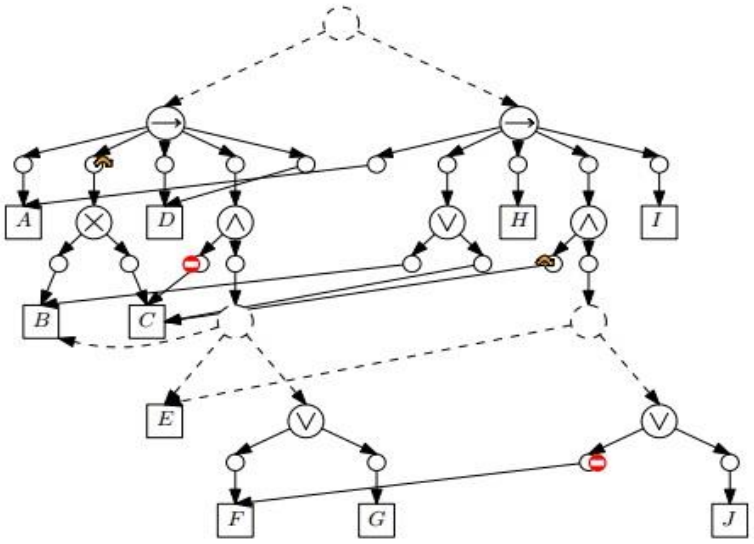


Figura 15: Enfoque CoSeNet.

En (Li, 2010) se discute un enfoque donde un modelo de proceso de referencia existente se mejora mediante el análisis de las diferentes variantes derivadas de ella. Sin embargo, el resultado no es un modelo de proceso configurable, sino solamente un modelo de proceso de referencia mejorada, es decir, las variantes se obtienen mediante la modificación del modelo de referencia en lugar de por la configuración del proceso.

En (La Rosa, 2012) se describe un enfoque alternativo que permite la fusión de los modelos de procesos en un modelo de proceso configurable, incluso si los modelos de procesos de entrada están en diferentes formalismos. En estos enfoques fusionados, algunas configuraciones pueden corresponder con un modelo de proceso defectuoso.

Otra forma de obtener un modelo de proceso configurable no es uniendo modelos de procesos, sino mediante la aplicación de técnicas de Minería de Procesos en una colección de registros de eventos, propuesto por (Gottschalk, 2008) donde se discutieron dos enfoques diferentes, pero no fueron soportados por los algoritmos de descubrimiento. El primer enfoque une a modelos de proceso descubiertos para cada registro de eventos usando técnicas para unir modelos de procesos existentes. En el segundo enfoque los registros de eventos se combinan primero y luego se combinan en un modelo de proceso que se descubre y se individualiza para cada registro de eventos.

En (Buijs, 2013) se proponen 4 enfoques para obtener un modelo de proceso configurable fusionando las diferentes variantes de modelos existentes (Figura 16, tomada (Buijs, 2013)).

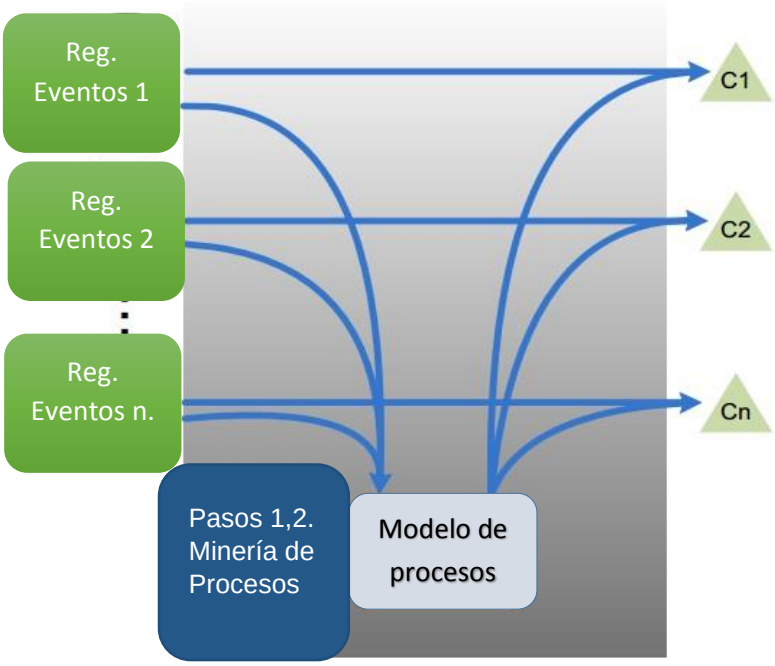


Figura 16: Modelo de procesos configurable.

Otra forma de representar los modelos de procesos configurables es mediante árboles de procesos configurables, que se construyen a través de los árboles de procesos.

1.7.1 Árbol de Procesos

En los lenguajes tradicionales como BPMN, Redes de Petri, Lenguaje Unificado de Modelado (UML, del inglés Unified Modeling Language), los diagramas de actividades podrían ser de la manera más conveniente para representar los modelos de procesos. Sin embargo, solamente una mínima parte de todos los modelos posibles en estos lenguajes son acertados, ejemplo, muchos modelos llegan a puntos muertos y otras anomalías. Es por ello que se elige usar árboles de procesos para describir los modelos (Buijs, 2012).

La Figura 17, tomada de (Buijs, 2012) muestra los posibles operadores en un Árbol de Procesos y su interpretación en una Red de Petri. Un Árbol de Procesos contiene nodos operadores y nodos hojas. Los nodos operadores especifican la relación entre sus hijos. Los posibles operadores son secuencia,

ejecución paralela, elección exclusiva, elección no exclusiva y ejecución de ciclo. El orden de los hijos importa para los operadores secuencia y ciclo. El orden de los hijos del operador de secuencia especifica el orden en el que se ejecutarán (de izquierda a derecha). Para el ciclo, el hijo izquierdo es el Redo, después de la ejecución del Do del hijo derecho, la parte Redo, puede ser ejecutada. Después de esta ejecución, la parte de Do puede ser habilitada de nuevo (Buijs, 2012).

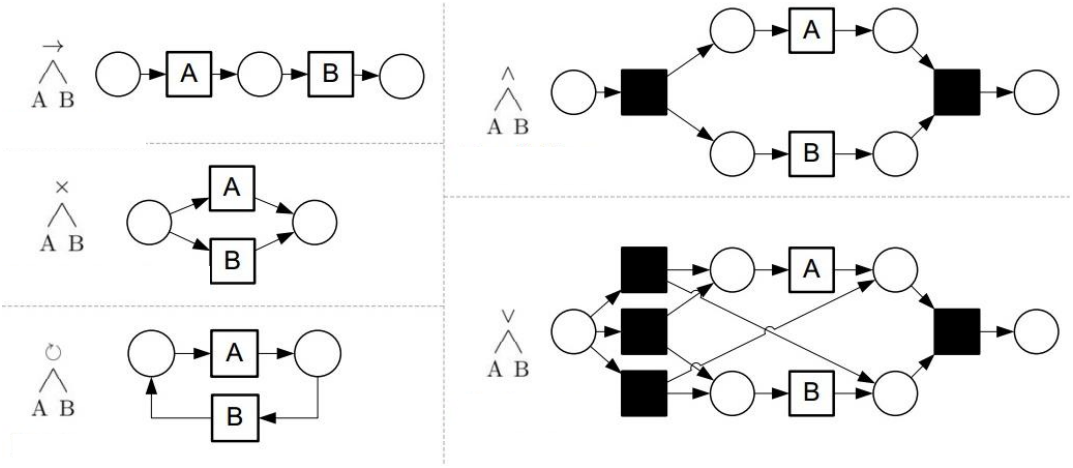


Figura 17: Posibles operadores en un Árbol de Procesos y su interpretación en una Red de Petri.

1.7.2 Árbol de Procesos Configurable

Según (Buijs, 2013) como se muestra en la Figura 18, tomada de (Buijs, 2013) los árboles de procesos configurables nos son más que distintos modelos de procesos llevados a la notación de árboles de procesos que mediantes opciones de configuración permiten las operaciones de bloquear y ocultar especificadas en los lenguajes de configuración existentes (Gottschalk, 2008), (Rosemann, 2007), (Schunselaar, 2012) donde bien se bloquea una ruta de ejecución u oculta una parte del proceso. Sin embargo, se añade la opción de configuración en la que los operadores pueden ser degradados. Por degradar un operador se entiende que el comportamiento del operador se limita a un subconjunto del comportamiento inicialmente posible. El operador ciclo por ejemplo, puede ser degradado a un operador secuencia. Esto se hace eliminando la parte Redo del ciclo (Buijs, 2013), es decir cuando aparezca el operador Ciclo se puede sustituir por un operador secuencia.

Otro operador que puede ser degradado es el elección no exclusiva que puede degradarse a un operador ejecución paralela forzando a todos los hijos a que sean ejecutados a la misma vez (Buijs, 2013), también

puede ser degradado a un operador elección exclusiva permitiendo solamente que un hijo sea ejecutado (Buijs, 2013), en este caso se le pide al usuario seleccionar el hijo que desea que se ejecute y por último se puede degradar a un operador secuencia ejecutando a todos los hijos en un orden particular (Buijs, 2013) brindado por el usuario.

Sin embargo, como en una configuración el orden de los hijos pudiera ser diferente a otra, se añade el operador secuencia inversa, que ejecuta los hijos en el orden inverso, es decir, de derecha a izquierda. Finalmente, también el operador ejecución paralela puede ser degradado a un operador secuencia o un operador secuencia inversa (Buijs, 2013).

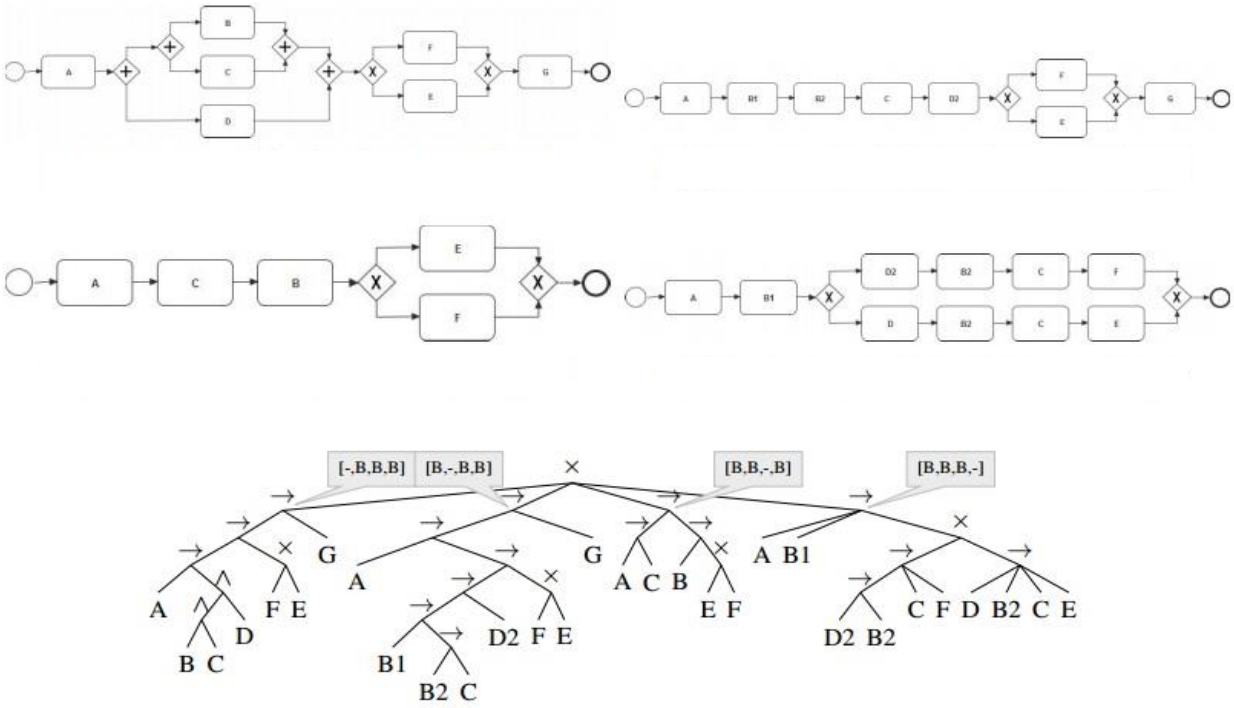


Figura 18: Árbol de Procesos Configurable.

1.8 Metodología de desarrollo

Una metodología es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información (GÓMEZ, y otros, 2010). Impone un proceso disciplinado sobre el desarrollo de software con

el objetivo de hacerlo predecible y eficiente (ALUR, y otros, 2003). Está compuesta por Tareas (actividades en que se dividen los procesos), Procedimientos (define la forma de ejecutar la tarea), Técnica (herramienta utilizada para aplicar un procedimiento), Herramientas (sistemas informáticos que automatizan la aplicación) y Producto (resultado de cada etapa) (Figueroa, 2011). Existen disímiles metodologías de desarrollo, todas ellas conforman dos grandes corrientes referentes a los procesos de desarrollo de software: las metodologías robustas o tradicionales y las metodologías ágiles, debido a que el equipo de desarrollo es pequeño y de corta duración los autores del trabajo se enfocan en las metodologías ágiles.

1.8.1 Metodologías ágiles

Las metodologías ágiles son apropiadas para guiar proyectos de poco volumen, que requieran una rápida implementación. Estas metodologías están especialmente indicadas en proyectos con requisitos poc**o** definidos o cambiantes, se aplican bien en equipos pequeños que resuelven problemas concretos (Figueroa, 2011).

Programación Extrema (XP, del inglés eXtreme Programing)

Es la más destacada de los procesos ágiles de desarrollo de software formulada por Kent Beck. La programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad.

Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos (Figueroa, 2011).

Las características fundamentales del método son (Figueroa, 2011):

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación.
- Programación por parejas.
- Frecuente interacción del equipo de programación con el cliente o usuario.

Proceso Ágil Unificado (AUP)

El Proceso Ágil Unificado (AUP, del inglés Agile Unified Process) es una versión simplificada del Proceso Unificado de Rational (RUP, del inglés Rational Unified Process) (Pressman, 1999), (Jacobson, 2000), que utiliza técnicas y conceptos de este. Desarrolla prototipos ejecutables durante la fase de elaboración del producto, demostrando la validez de la arquitectura para los requisitos clave del producto y determinando los riesgos técnicos. AUP es flexible y propone los mismos roles y artefactos que RUP, solo que no hay necesidad de generar toda la documentación que se requiere en cada flujo de trabajo.

Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. Aplica técnicas ágiles incluyendo Desarrollo Dirigido por Pruebas.

AUP mantiene las cuatro fases propuestas por RUP (inicio, elaboración, construcción y transición), no así para los flujos de trabajo donde el Modelo, primer flujo establecido por AUP, abarca las disciplinas: Modelo de Negocio, Requerimientos y Análisis y Diseño (Figura 19, tomada de (J. L. Cordero, 2014)).

Fases:

- Inicio: Se determina el alcance inicial del proyecto, se identifica una arquitectura candidata para el sistema y se obtiene la financiación inicial del proyecto así como la aceptación de los interesados.
- Elaboración: Se prueba la arquitectura del sistema.
- Construcción: Se construye un software de manera regular y gradual partiendo por aquellas funcionalidades que poseen la prioridad más alta en términos de necesidad del cliente.
- Transición: Se valida y despliega el sistema en el entorno de producción (J. L. Cordero, 2014).

Flujos de Trabajo:

- Modelo: Este flujo de trabajo centra su atención en comprender los procesos de negocio que tienen lugar dentro de la organización, el problema inicial que dio origen al proyecto así como identificar una solución viable para el entorno de la empresa.
- Implementación: El objetivo de este flujo de trabajo es transformar los modelos en código ejecutable y desarrollar un nivel básico de pruebas, particularmente pruebas de unidad.

- Prueba: Se persigue desarrollar un proceso de evaluación de software que garantice la calidad del producto a través de la detección de errores, comprobar que el sistema funcione de acuerdo a su diseño y que se hayan cumplido los requisitos funcionales y no funcionales del mismo.
- Despliegue: Se establece un plan para le entrega del sistema de manera que los usuarios finales reciban un software completamente funcional.
- Administración de Configuración: Se administra el acceso a los artefactos producidos como resultado del proceso de desarrollo del software, así como la actualización y el control de versiones.
- Administración de Proyecto: La finalidad de este flujo consiste en dirigir las actividades que tienen lugar dentro del proyecto, esto incluye administración de riesgos y dirección del personal, así como la coordinación entre personas y sistemas fuera del marco del proyecto con el objetivo de garantizar la entrega del sistema en tiempo y sin exceder el presupuesto.
- Ambiente: Dispone actividades que describen los procesos y herramientas (hardware, software, etc.) que soportará el equipo de trabajo del proyecto (J. L. Cordero, 2014).

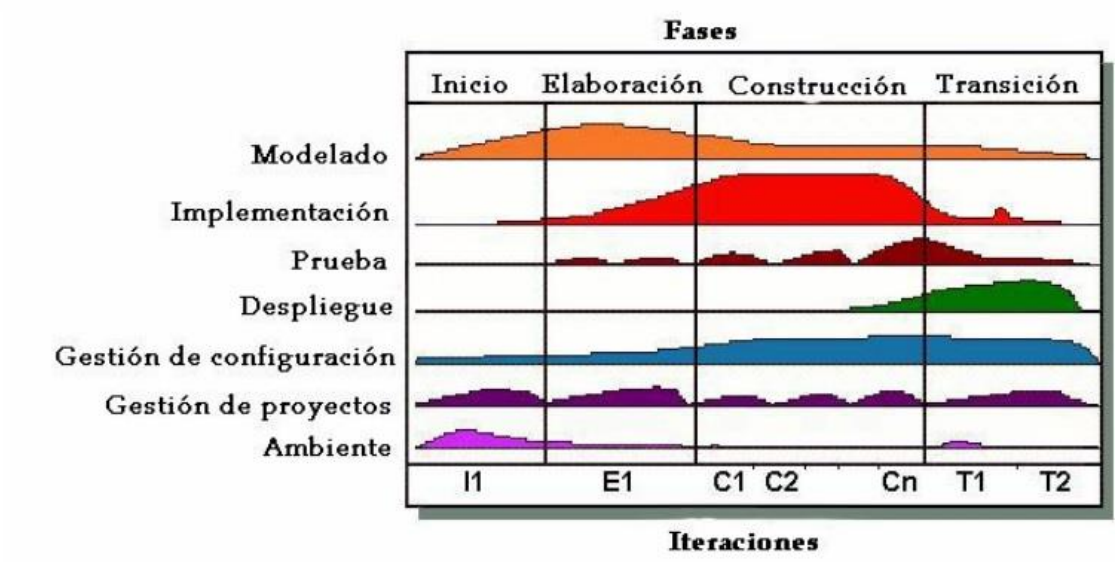


Figura 19: Flujos de trabajo y fases de AUP.

1.8.2 Justificación de la metodología seleccionada

La metodología que se elige para llevar a cabo el desarrollo de la aplicación es AUP, debido a que se basa en la metodología RUP estudiada a lo largo de la carrera y que en su versión ágil permite simplificar

la cantidad de artefactos y el tiempo de desarrollo adecuándose a las características del equipo de proyecto, el cual es un equipo pequeño y de corta duración. AUP es una metodología flexible que no requiere de una gran cantidad de desarrolladores, se centra en actividades de alto valor esenciales para el desarrollo del producto, no necesita herramientas específicas para trabajar, usándose las que disponga el usuario y es de fácil adaptación. Es concisa en el aspecto de la documentación, permitiendo generar solo la necesaria y no la especificada para cada flujo de trabajo como lo hace RUP. No se podría elegir XP, porque el equipo de desarrollo no tiene experiencia en el trabajo con esta metodología y la misma precisamente se basa en la capacidad y madurez de los miembros del equipo. La metodología RUP serviría si se dispusiera de más tiempo para el desarrollo del sistema.

1.9 Lenguajes y Herramientas

Previo al diseño e implementación del complemento, se indagó, tanto dentro del ámbito nacional como internacional, sobre herramientas que pudiesen brindar las funcionalidades necesarias para darle solución al objetivo planteado en la investigación.

Marco de trabajo

El marco de trabajo (del inglés framework) de desarrollo que se utilizará es ProM en su versión 6.2. Constituye un marco de trabajo extensible que provee un conjunto de herramientas en forma de complementos para el descubrimiento y análisis de modelos de procesos a partir de registros de eventos. Está implementado en el lenguaje Java, es de código abierto y es multiplataforma, adaptado para dar soporte a complementos de Minería de Procesos (Aalst, 2010). Esta herramienta contiene una amplia variedad de complementos que implementan algoritmos para obtener y almacenar los modelos, importar objetos, analizar y convertir los resultados (Günther, 2009), (Aalst, 2007).

Es compatible con los estándares de registros de eventos MXML (del inglés, Mining eXtensible Markup Language) y XES (del inglés, eXtensible Event Stream). Los complementos que se desarrollan en la actualidad que implementan los nuevos algoritmos y técnicas, se añaden a este sistema creado por el equipo líder en esta ciencia. Por eso se ha convertido en la herramienta por excelencia para la Minería de Procesos.

Lenguaje de programación

El lenguaje de programación a utilizar será Java versión 7, que es orientado a objetos, ofrece la posibilidad de escribir el código una sola vez y ejecutarlo en múltiples entornos gracias a lo que se conoce como bytecode, que no es más que código interpretado por una Máquina Virtual de Java o JVM específica para cada entorno. Este bytecode solo debe ser generado una vez y puede ser interpretado por todas las JVM homologadas.

Entorno de desarrollo integrado

Como entorno de desarrollo integrado (IDE por sus siglas del inglés) se utilizará el Eclipse versión 1.4 ya que posee un entorno integrado de código abierto y es multiplataforma. Dispone de un editor de texto con resaltado de sintaxis; compilación en tiempo real y tiene la capacidad de expandir su potencial utilizando módulos. Además de que el código fuente de los paquetes del ProM, que están disponibles en el repositorio de Internet son proyectos desarrollados en Eclipse.

Lenguaje Unificado de Modelado

La metodología AUP propone utilizar el lenguaje UML para especificar, visualizar y construir los artefactos de los sistemas de software. Está destinado a los sistemas de modelado que utilizan conceptos orientados a objetos (Larman, 1999).

Para el modelado UML existen varias aplicaciones, estas constituyen las herramientas CASE. Una herramienta CASE se define como un producto basado en computadora destinado a apoyar a una o más actividades de ingeniería de software en un proceso de desarrollo de software (DIXON, 2004). La principal ventaja de la utilización de estas herramientas es el aumento de la productividad y la mejora de la calidad del producto final.

Visual Paradigm

Visual Paradigm es una herramienta de Ingeniería de Software Asistido por Computadoras (CASE, del inglés Computer Aided Software Engineering) multiplataforma, que permite el modelado UML, y soporta el ciclo de vida completo de RUP, permitiendo generar casi cualquier documento o diagrama. Puede integrarse con entornos integrados de desarrollo como el Eclipse y el NetBeans. Permite hacer diagramas UML, generación automática de código, sincronización entre modelos y código, entre otras posibilidades. Se utilizará el Visual Paradigm en su versión 8.0 (PARADIGM, 2012).

1.10 Conclusiones del capítulo

En este capítulo se analizaron conceptos relacionados con la Gestión de Procesos de Negocio, la Minería de Procesos, sus tipos, las técnicas de diagnóstico de procesos existentes, las notaciones existentes para representar los modelos de procesos, así como un estudio de algunas de las metodologías de desarrollo de software más utilizadas en la actualidad.

Este estudio permitió comprender que una forma de entender un proceso poco estructurado es convertirlo en una familia de procesos mediante una separación de variantes. Existe una técnica que consiste en una representación de diferentes descomposiciones en subprocesos pero su limitación es que esta no visualiza todas las variantes de un proceso simultáneamente. Por ello se decidió usar para este trabajo la técnica Árbol de Procesos Configurable que permitirá mostrar todas las variantes del proceso simultáneamente y poderlo configurar posteriormente. El análisis de los modelos de procesos configurables en las diferentes notaciones existentes permitió seleccionar el Árbol de Procesos como notación para la técnica a desarrollar, debido a que estructuralmente es la más parecida a Árbol de Variantes por su forma jerárquica. Se fundamentó la selección del lenguaje java, la herramienta CASE Visual Paradigm, el entorno de desarrollo eclipse y el marco de trabajo colaborativo ProM, como herramientas para el desarrollo de la propuesta. Por último, la metodología seleccionada para el desarrollo de la investigación fue RUP Ágil, debido a que es la más adecuada para las condiciones en las que se desarrolla la investigación.

Capítulo 2: Propuesta de solución

2.1 Introducción

En el presente capítulo se presenta la propuesta de solución de la investigación. Para ello se describe el proceso de negocio, se definen y describen los requisitos funcionales y no funcionales. Además, se detalla el diseño de la solución a través de artefactos como el diagrama de paquetes, de clases, de despliegue, de componentes, utilizando patrones de diseño y métricas para validarlo.

2.2 Descripción general de la propuesta del sistema

Para dar cumplimiento a los objetivos previamente planteados, el sistema propuesto mostrará un Árbol de Procesos Configurable a partir de un Árbol de Variantes como entrada, y responderá a las acciones de bloquear, degradar y ocultar, haciendo más entendible el modelo obtenido, brindando además la posibilidad de reubicar los nodos de tal forma que el modelo sea más manejable por el analista.

2.2.1 Modelo de dominio

Un modelo de dominio describe y relaciona los conceptos importantes del contexto en forma de objetos del dominio. La identificación y asignación de un nombre para estos objetos ayudan a desarrollar un glosario de términos que permitirán comunicarse mejor a todos los que están trabajando en el sistema (Jacobson, 2000).

El modelo de dominio se describe mediante diagramas UML (principalmente mediante diagramas de clases). Estos diagramas le muestran a los clientes, usuarios, revisores y a otros desarrolladores las clases del dominio y cómo se relacionan unas con otras mediante asociaciones, además de ayudar comprender los conceptos claves del ámbito del problema a resolver.

En la Figura 20 se muestra el modelo del dominio conformado en la investigación:

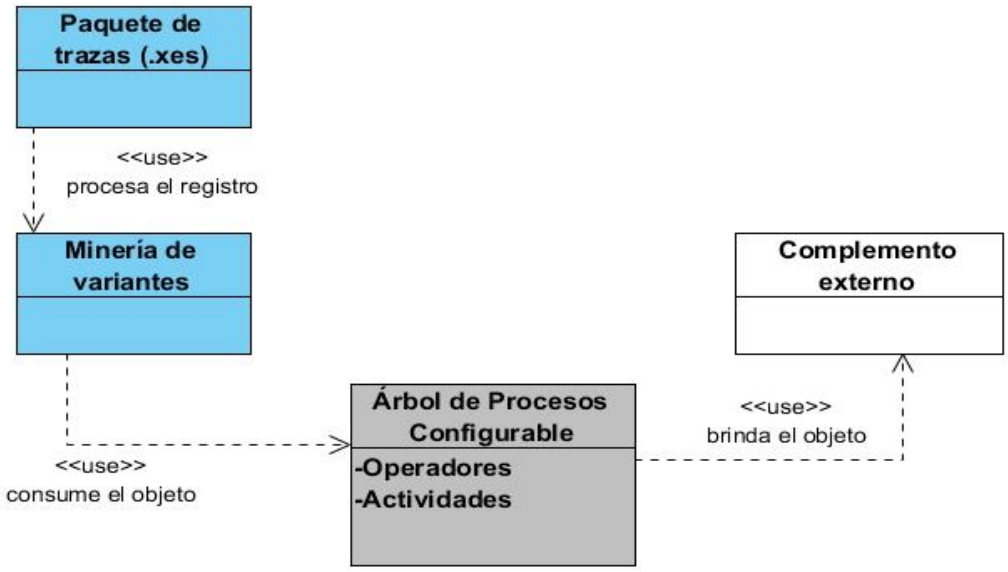


Figura 20: Modelo de dominio.

2.3 Requisitos

Un requisito es un atributo necesario en un sistema, una instrucción que identifica una capacidad de, típico, o factor de calidad de un sistema con el fin de que tenga valor y la utilidad de un cliente o usuario (Pressman, 2005).

Técnicas para capturar los requisitos

Las técnicas para la captura de requisitos sirven para entender cuáles son las principales características que debe tener el sistema y recopilar el conocimiento sobre los requisitos del mismo. Algunas de estas técnicas que fueron empleadas son:

Sistemas existentes: Se realizó un análisis de las técnicas de diagnóstico existentes para ver las limitaciones de las mismas y determinar características y funcionalidades que se le pudieran mejorar en la presente investigación.

Revisión de documentos: Se consultaron diferentes documentos existentes sobre las técnicas y algoritmos de descubrimientos de procesos actuales permitiendo un entendimiento más amplio sobre las diferentes funciones que desempeñan y lograr así una correcta identificación de los requisitos que fueron implementados.

Tormenta de ideas: Se realizaron reuniones de grupos de trabajo puede estar o no involucrado el cliente final del producto.

Para el levantamiento de los requisitos de la solución a desarrollar se tuvieron en cuenta estas tres técnicas abordadas anteriormente.

Requisitos funcionales

Los requerimientos funcionales definen las funciones que el sistema será capaz de realizar. Describen las transformaciones que el sistema realiza sobre las entradas para producir salidas (Pressman, 2005). A continuación se presentan los requisitos funcionales detectados.

1. Mostrar múltiples variantes del proceso contenidas en un Árbol de Procesos Configurable.
2. Exportar el Árbol de Procesos Configurable.
3. Degradar Árbol de Procesos Configurable.
4. Ocultar nodos.
5. Bloquear nodos.
6. Visualizar un Árbol de Procesos Configurable.
7. Reubicar los nodos dentro del área de trabajo.
8. Visualizar un Árbol de Procesos Configurable a partir de una nueva configuración del Árbol de Procesos.
9. Deshacer la acción bloquear o degradar.
10. Deshacer la acción ocultar.
11. Importar el árbol.

Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. En muchos casos los requisitos no funcionales son fundamentales en el éxito del producto. Normalmente están vinculados a requisitos funcionales y son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que el mismo cumple con toda la funcionalidad requerida, las propiedades no funcionales, pueden marcar la diferencia entre un producto

bien aceptado y uno con poca aceptación (Pressman, 2005). A continuación se presentan los requisitos no funcionales definidos para la aplicación:

Requisitos de usabilidad:

La interfaz debe asegurar que el usuario comprenda intuitivamente las funciones que puede realizar en ella.

Requisitos de portabilidad:

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 diseño e implementación:

El lenguaje de programación deberá ser java para estar acorde con el mismo lenguaje en que se ha desarrollado el ProM, y utilizar la tecnología libre.

Requisitos de software y hardware:

Se recomienda utilizar como sistema operativo Windows 7, o preferiblemente una distribución de software libre, tener instalada la máquina virtual de java y tener algunos paquetes del ProM, en especial el VariantMiner (Minería de Variantes). Además reservar para el sistema 1 GB de memoria RAM y 2 GB para el disco duro.

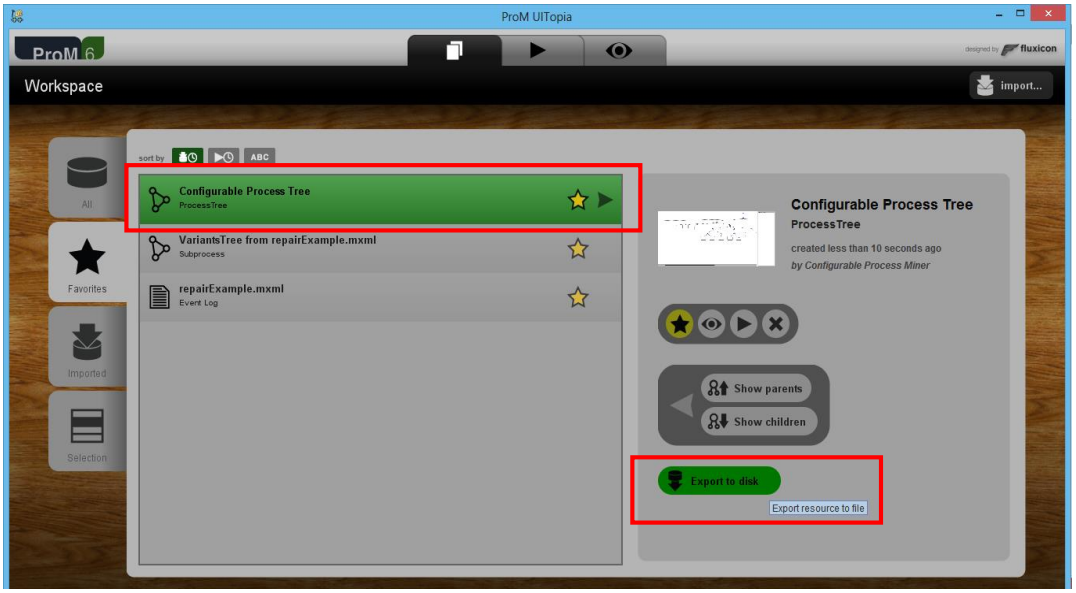
Soporte

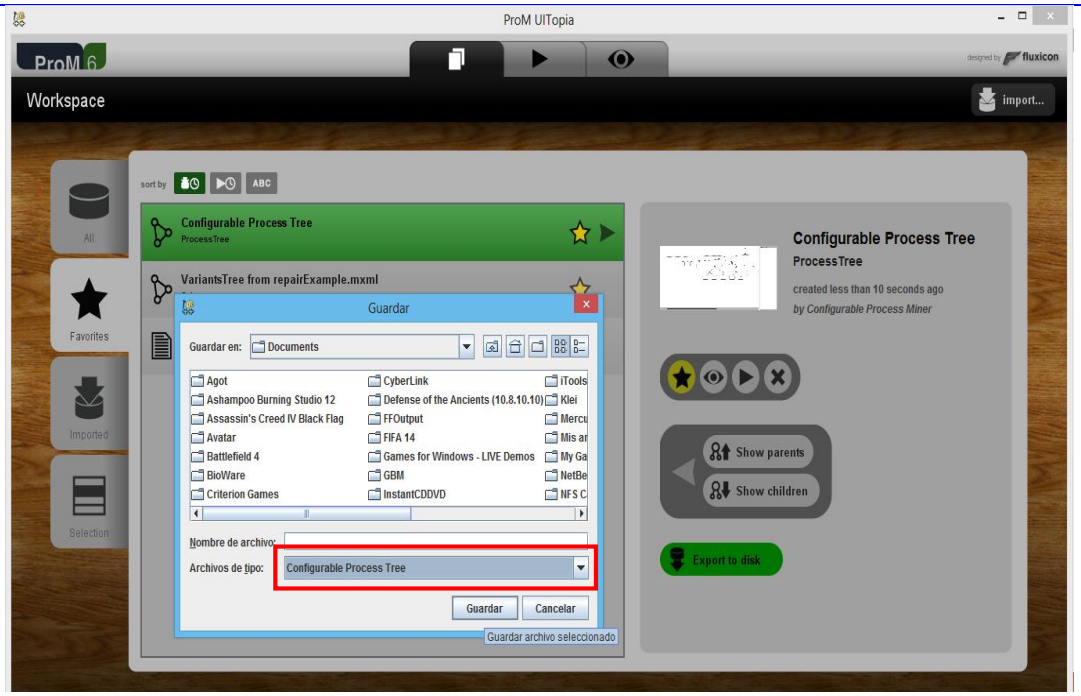
La aplicación contará antes de su puesta en marcha con un período de pruebas, se le dará mantenimiento, configuración y se brindará el servicio de instalación. Ver documento "*Pasos para instalar el ProM 6.2"*.

Especificación de requisitos

La especificación de los requisitos es una descripción detallada del comportamiento del sistema que se va a desarrollar. Entre los requerimientos más significativos se encuentran: visualizar todas las variantes de forma simultánea en el Árbol de Procesos Configurable, exportar el Árbol de Procesos Configurable, degradar Árbol de Procesos Configurable y visualizar un Árbol de Procesos Configurable a partir de una

configuración contenida en un Árbol de Procesos**.** La especificación de los requisitos se encuentra descrita en el documento *"Especificación de Requisitos de Software del complemento para generar un Árbol de Proceso Configurable"*. A continuación se muestra la especificación del requisito Exportar el Árbol de Procesos Configurable.

Nº	**Nombre**	**Descripción**	**Complejidad**	**Prioridad para cliente**
RF02	Exportar el Árbol de Procesos Configurable.	Permite exportar el árbol como un archivo de tipo Árbol de Procesos Configurable.	Media	Media
	Prototipo			
				



Campos	Tipos de Datos	Reglas o Restricciones
Nombre del archivo	Cadena de caracteres	Obligatorio Longitud: Máximo 250 caracteres Formato: Comienza con mayúscula
Tipo	Nomenclador	Obligatorio
Observaciones		

En general, se determinó la complejidad de todos los requisitos utilizando la técnica de estimación propuesta por el Programa de Mejora del Centro CEGEL, donde se definen una serie de indicadores a

medir, como es el caso de complejidad por interfaces, diferentes comportamientos, formas de inicialización, consultas a fuentes de almacenamientos, restricciones de validación, grado de reutilización y lógica de negocio, los cuales se encuentran descritos en el documento ”*Evaluación de Requisitos del Complemento para generar un Árbol de Procesos Configurable, sección Guía Complejidad”.*

A continuación se muestra la cantidad de requisitos que presentan baja, media y alta complejidad:

• 5 de complejidad baja.

• 2 de complejidad media.

• 4 de complejidad alta.

2.3.1 Métricas para la validación de 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, 2010).

Para la validación de los requisitos de la solución propuesta se decidió la utilización de cuatro técnicas principales:

- Estabilidad de los requisitos.
- Grado de validación de los requisitos.
- Especificidad de los requisitos.
- Prototipos de interfaz de usuario.

Para medir la Estabilidad de los requisitos se aplicó la técnica de la siguiente manera:

De los once requisitos funcionales que se identificaron, dos fueron sometidos a modificaciones luego de un análisis realizado con el cliente. Tomando ese número de modificaciones (2) como las variables de entrada para la fórmula que se define a continuación:

$$ETR = \frac{[RT - RM]}{RT} * 100$$

ETR = [11-2/11] * 100 = 81,8

Donde:

ETR: Estabilidad de los requisitos.

RT: Total de requisitos definidos.

RM: Cantidad de requisitos modificados.

La técnica arrojó como resultado un 81% de estabilidad, valor realmente alto por lo se llegó a la conclusión de que los requisitos definidos son estables, asegurándose así que es confiable trabajar el análisis y diseño sobre ellos. A continuación se muestra una gráfica (Figura 21) que muestra la estabilidad de los requisitos.

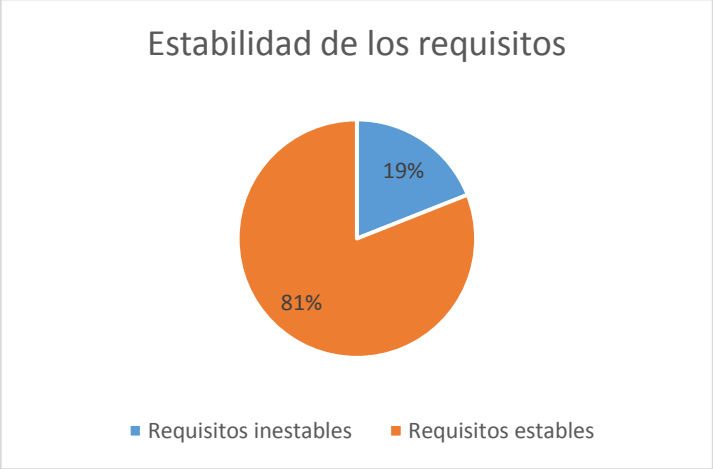


Figura 21: Estabilidad de los requisitos.

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

$$VR = \frac{nc}{(nc + nnv)}$$

VR = 11/(11+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.

Para cuantificar la Especificidad de los requisitos se aplicó la técnica de la siguiente manera:

$$Q1 = \frac{nui}{nr}$$

Q1 = 11/11 = 1.

Donde:

Q1: Especificidad de los requisitos.

Nr: Total de requisitos definidos.

Nui: Cantidad de requisitos para los que los revisores tuvieron interpretaciones idénticas.

Por lo tanto se puede asegurar que los requisitos no tienen ambigüedad y por tanto la especificación de requisitos se hizo con calidad.

Por último y no menos importante se utilizó la técnica prototipos de interfaz de usuario, permitiendo la obtención de prototipos a partir de la definición de requisitos que, sin tener la totalidad de la funcionalidad del sistema, ayudaron al usuario hacerse una idea de la estructura de la interfaz del sistema. *Ver (prototipos de interfaz de usuario).*

2.3.2 Cómo convertir el producto final en un paquete para el ProM

Usar el complemento sin necesidad de ejecutarlo desde el Eclipse es una de las funcionalidades que brinda el marco de trabajo ProM. Solo se necesita realizar una serie de pasos que hacen que la aplicación funcione y pueda cargarse como un paquete del mismo y además subirse al repositorio en Internet donde se encuentran los proyectos desarrollados para el ProM. Para obtener el complemento como paquete se deben seguir los siguientes pasos:

1- Exportar el proyecto de eclipse como archivo .jar.
2- Copiar en C:\Users\....\.ProM62\packages, dentro de una carpeta con el nombre del complemento el archivo .jar.
3- Abrir el packages.xml y poner en paquetes instalados <package name="…" version="…" os="…" url="C:\Users\carlos\.ProM62\packages\prueba\cpt.jar" desc="prueba for Windows 64bit" org="SourceForge" auto="false" hasPlugins="true" license="unknown" author="Carlos, LpSolve, peno64" maintainer="Daniel, LpSolve, peno64" logo=""> </package>.
4- También se deben escribir las dependencias del complemento si tiene.
5- La url debe apuntar al propio complemento para que luego se pueda instalar desde el Administrador de paquetes que se encuentra en la carpeta del mismo ProM.
6- Ejecutar el ProM.

2.4 Diseño de la solución

En este epígrafe se pueden observar los diferentes diagramas de clases realizados para el diseño, la aplicación de los patrones GRASP (Patrones para la asignación de responsabilidad), los patrones GoF y la aplicación de métricas para validar el diseño.

2.4.1 Diagrama de paquetes

El diagrama de paquetes estructura las clases de una manera más organizada en paquetes, para poder comprender mejor los diagramas de clases que cuentan con grandes cantidades de clases. La Figura 22 muestra la estructuración por paquetes conformada a partir de las clases del diseño.

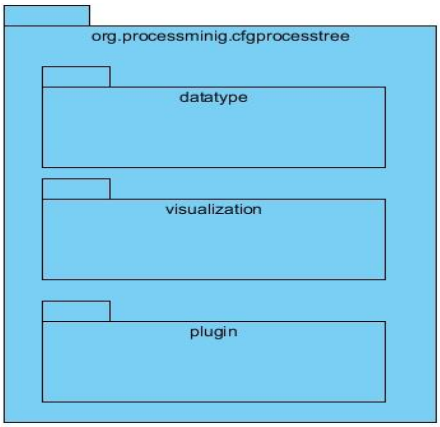


Figura 22: Diagrama de paquetes.

2.4.2 Diagrama de clases

El Diagrama de clases ilustra una vista general de las principales clases utilizadas en el diseño, sus atributos y métodos, así como la relación que existe entre ellas. En la Figura 23 se muestran las clases que implementan las interfaces del paquete datatype.

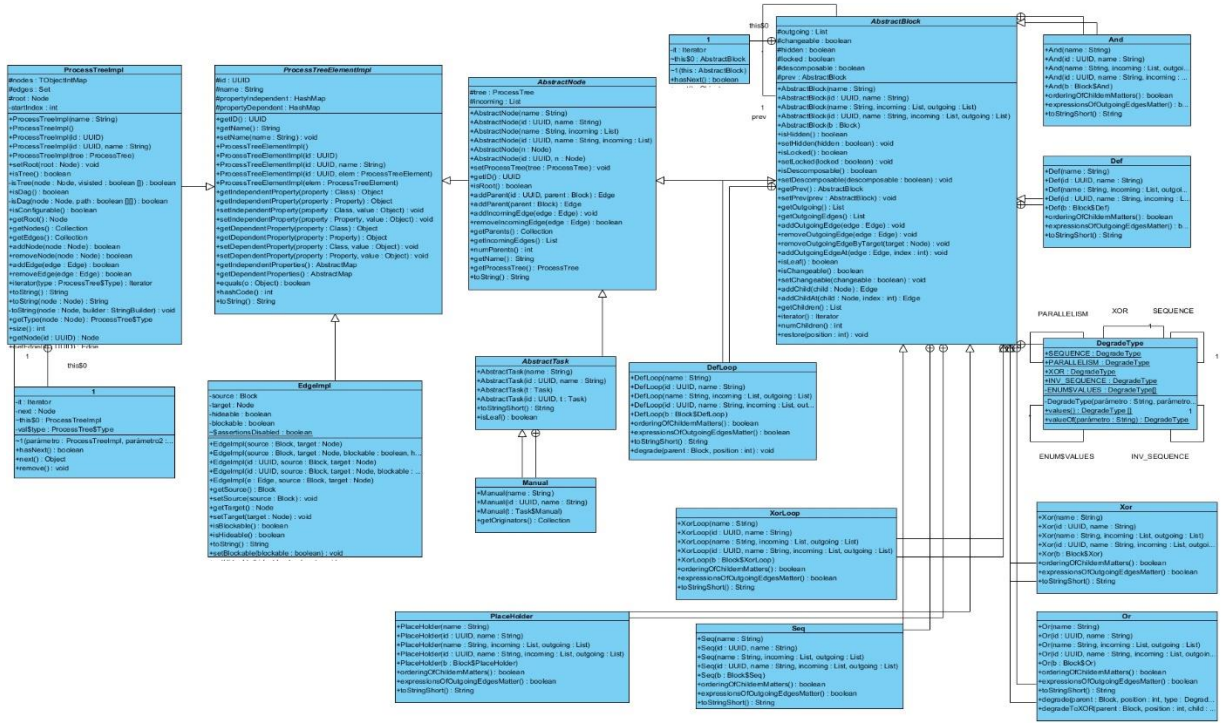


Figura 23: Clases del paquete datatype.

2.4.3 Patrones de diseño

Para elaborar el diseño se utilizan un grupo de patrones o modelos que ayudan en el cumplimiento de los objetivos de la solución. Los Patrones de Diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces (Patrones GRASP, 2011).

Para definir el diseño de la solución propuesta se tuvieron en cuenta varios patrones, a continuación se especifican los seleccionados:

Patrones GRASP

GRASP es el acrónimo de General Responsibility Assignment Software Patterns (Patrones de asignación de responsabilidades). Los GRASP son patrones para la asignación de responsabilidades, que ayudan a entender el diseño de objetos (Larman, 1999).

Experto: Se pone en práctica con el uso de clases que poseen responsabilidades específicas a cumplir, de acuerdo con la información que manejan. El uso de este patrón se evidencia en la clase TreeVisualization (Figura 24) que posee funciones concretas de acuerdo con la información que gestiona.

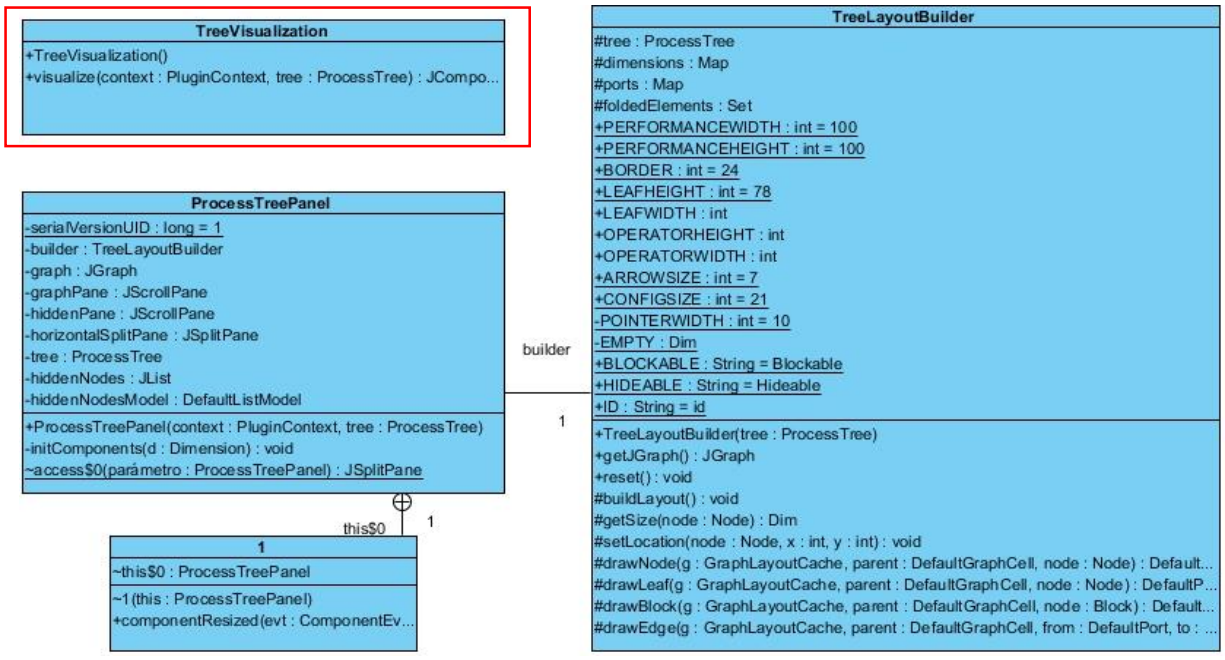


Figura 24: Clases del paquete tree.

Alta Cohesión: Se refiere a cuán relacionadas y enfocadas están las relaciones de una clase, por lo que una clase con alta cohesión tiene responsabilidades estrechamente relacionadas y poco complejas. Un ejemplo de este patrón se evidencia en las clases ConfigurableProcessTreeImportPlugin y ConfigurableProcessTreeExportPlugin.

Bajo Acoplamiento: Este patrón se refiere a cuán fuertemente está relacionada una clase con otra, de forma tal que si se produce una modificación entre sus relaciones, se afecten en menor medida el resto de las clases, por lo que una clase con bajo acoplamiento no depende de muchas otras para realizar su función en el negocio. Un ejemplo de este patrón se evidencia en la clase ConfigurableProcessTreePlugin y en la mayoría de las clases del sistema, ya que se relacionan a través de interfaces que brindan las librerías del ProM y no con otras clases en específico.

Controlador: Este patrón sugiere el uso de clases controladoras para separar la lógica de negocios de la capa de presentación. Las peticiones en la aplicación son manejadas por clases de este tipo, por lo que no existe una sola clase controladora que se encargue de manejar todas las peticiones. Un ejemplo de esta clase es la ConfigurableProcessTreeManager.

Estos tres patrones se pueden observar en la Figura 25.

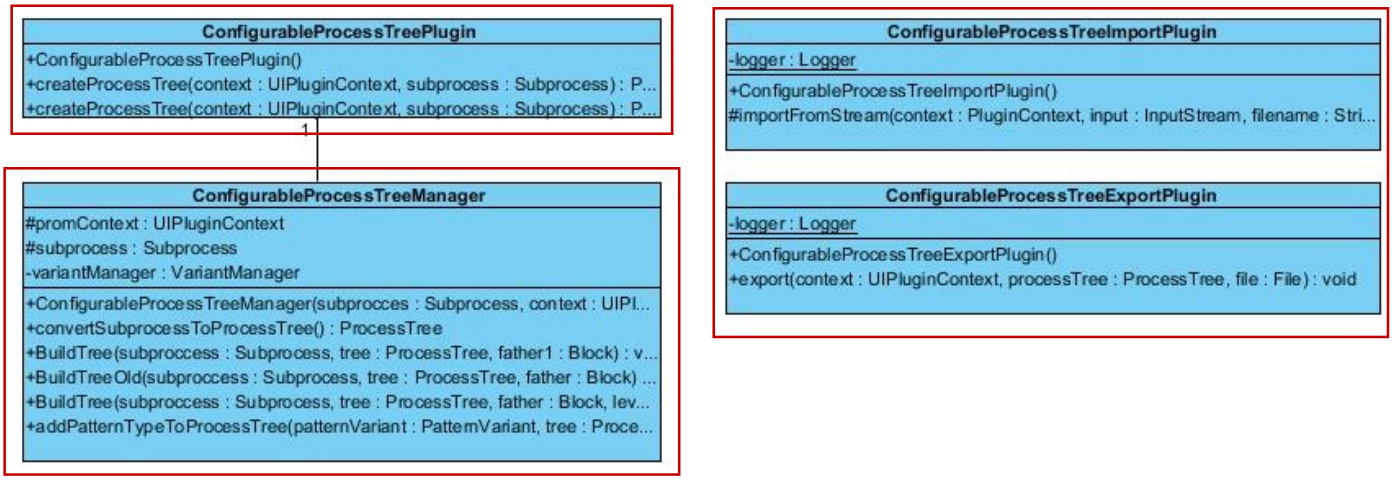


Figura 25: Clases del paquete plugin.

Patrones GoF

Los patrones de diseño GoF se clasifican en 3 categorías basadas en su propósito: creacionales, estructurales y de comportamiento (Madrid, 2005).

1. Creacionales: Tratan con las formas de crear instancias de objetos. El objetivo de estos patrones es abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.
2. Estructurales: Describen cómo las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos.
3. Comportamiento: Ayudan a definir la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos.

Se usaron varios patrones de las diferentes categorías, a continuación se explican:

Estructurales:

Composición: Patrón que trata uniformemente a los objetos simples y compuestos de una estructura jerárquica recursiva. Se usa cuando se pretende representar una jerarquía recursiva de objetos. Favorece la extensibilidad, ya que es muy fácil añadir nuevos tipos de componentes. Se usó durante el desarrollo de la aplicación en la clase ProcessTreeImpl (Figura 26).

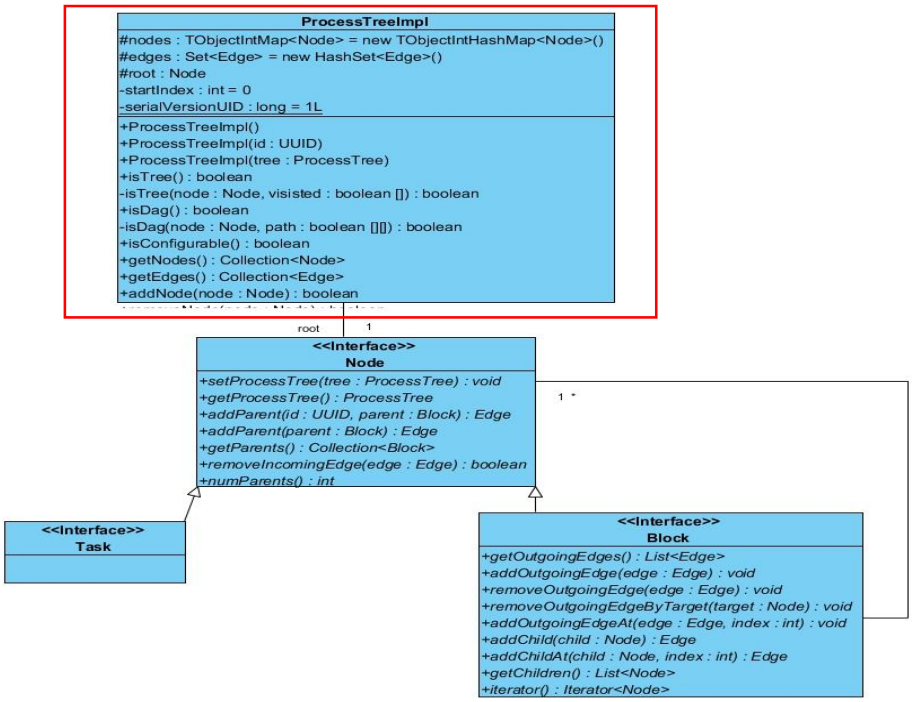


Figura 26: Ejemplo del patrón Composición.

Comportamiento:

Iterador: Este patrón proporciona una forma de acceder secuencialmente a los elementos de un objeto compuesto por agregación sin necesidad de desvelar su representación interna. Se usa cuando se quiere proporcionar una interfaz uniforme para recorrer diferentes estructuras de agregación. Se emplea en algunas de las clases del paquete visualization a la hora de conformar el árbol, ver Figura 24.

Solitario: Este garantiza que solamente se cree una instancia de la clase y le proporciona un punto de acceso global. Utilizando este patrón se reduce el espacio de nombres (frente al uso de variables globales), permite refinamientos en las operaciones y en la representación, mediante la especialización por herencia de "Solitario". Además, es fácilmente modificable para permitir más de una instancia y, en general, para controlar el número de las mismas (incluso si es variable). Se pone de manifiesto en la clase Subprocess (Figura 27) usándose en algunas clases como ConfigurableProcessTreeManager.

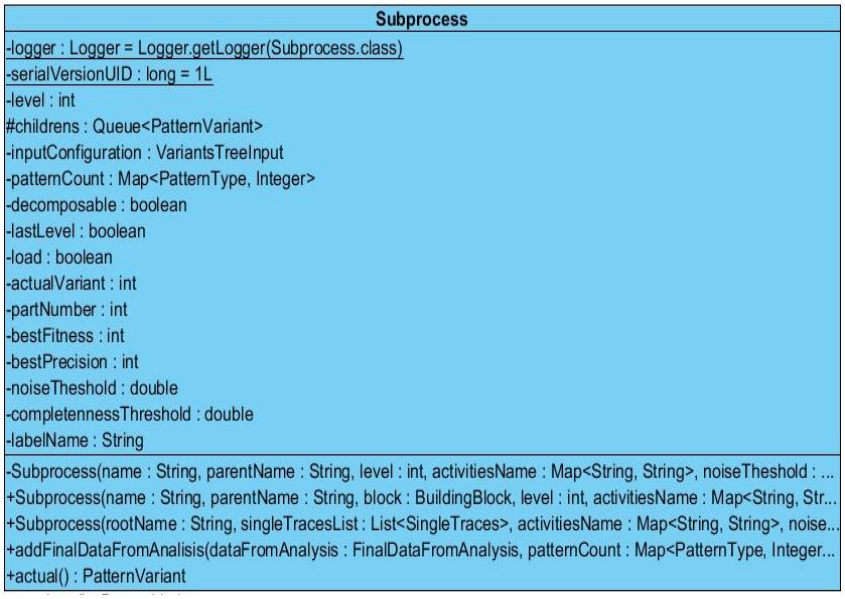


Figura 27: Ejemplo del patrón Solitario, clase Subprocess.

2.4.4 Métricas para validar el diseño

Las métricas de software son una medida cuantitativa del grado en que un sistema, componente o proceso, posee un atributo dado. Permiten averiguar cuán bien están definidas las clases y el sistema, lo que impacta directamente en el mantenimiento del mismo, tanto por la comprensión de lo desarrollado

como por la dificultad de modificarlo con éxito. Es decir, estas métricas permiten medir de forma cuantitativa la calidad de los atributos internos del producto, lo que le permite al ingeniero evaluar la calidad durante el desarrollo del sistema. Estas métricas tienen como propósito entender y mejorar la calidad del producto, evaluar la efectividad del proceso y mejorar la calidad del trabajo llevado a cabo al nivel del proyecto (Pressman, 2005).

Son varios los puntos de vista relacionados con la calidad del software. Las métricas de diseño a nivel de componentes se concentran en las características internas de los componentes del software con medidas que pueden ayudar al desarrollador a juzgar la calidad de un diseño a nivel de componente (Pressman, 2005).

Se utilizaron las métricas Tamaño Operacional de Clase (TOC) y Relaciones Entre Clases (RC), con el objetivo de evaluar la calidad del diseño propuesto. Estas métricas posibilitan medir los siguientes atributos de calidad (Kidd, 1994):

- Responsabilidad: Responsabilidad que posee una clase en un marco de modelado de un dominio correspondiente al modelado de la solución propuesta.
- Complejidad del mantenimiento: Es el grado de esfuerzo necesario para sustentar, mejorar o corregir el diseño de software propuesto. Puede influir significativamente en los costes y la planificación del proyecto.
- Complejidad de implementación: Grado de dificultad que posee implementar un diseño de clases determinado.
- Reutilización: Es cuán reutilizada puede ser una clase o estructura de clase dentro de un diseño de software.
- Acoplamiento: Dependencia o interconexión de una clase o estructura de clase con otras.
- Cantidad de pruebas: Número o grado de esfuerzo necesario para realizar las pruebas de calidad al producto diseñado.

A continuación se detalla la manera en que se evalúan los atributos de calidad de la métrica TOC.

Tamaño Operacional de Clase (TOC)

Está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad:

Tabla 1: Atributos de calidad evaluados por la métrica TOC.

Atributo de calidad	Modo en que lo afecta
Responsabilidad	Aumento del TOC provoca aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Aumento del TOC provoca aumento de la complejidad de implementación de la clase.
Reutilización	Aumento del TOC provoca disminución del grado de reutilización de la clase.

En la siguiente tabla se detalla la manera en que se evalúan los atributos de calidad de la métrica TOC.

Tabla 2: Criterios de evaluación para la métrica TOC.

Atributo	Categoría	Criterio
Responsabilidad	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Complejidad de implementación	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio

A continuación se detalla la manera en que se evalúan los atributos de calidad de la métrica RC.

Relaciones entre Clases (RC)

Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad:

Tabla 3: Atributos de calidad evaluados por la métrica RC.

Atributo de calidad	Modo en que lo afecta
Acoplamiento	Aumento del RC provoca aumento del Acoplamiento de la clase.
Complejidad de mantenimiento	Aumento del RC provoca aumento de la complejidad del mantenimiento de la clase.
Reutilización	Aumento del RC provoca disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Aumento del RC provoca aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Para evaluar el diseño del presente trabajo se definieron los siguientes criterios y categorías de evaluación para los atributos de calidad anteriores:

Tabla 4: Criterios de evaluación para la métrica RC.

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de mantenimiento	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio
Cantidad de pruebas	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio

Resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad.

Como resultado de la evaluación de la métrica TOC se obtuvo lo siguiente (Figura 288 y Figura 299):

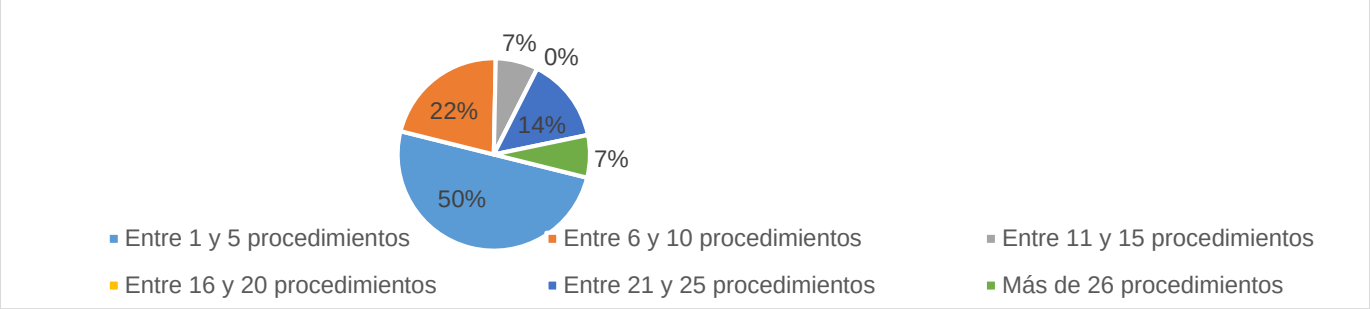


Figura 28: Resultados de la evaluación.

Representación en porciento de la incidencia de los resultados obtenidos en cada atributo de calidad:

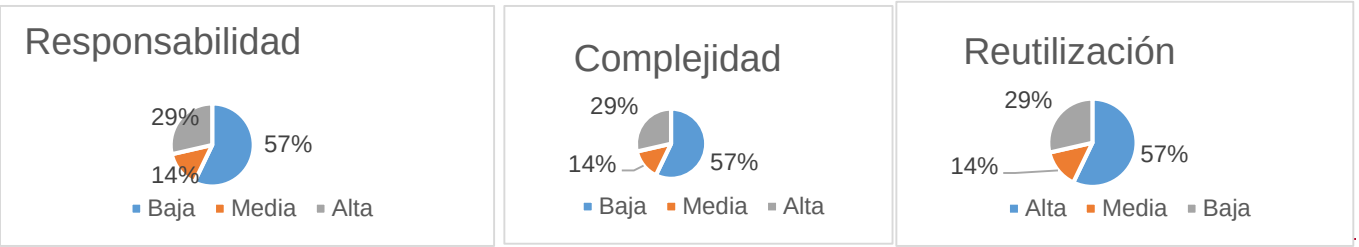


Figura 29: Representación en porciento de los resultados.

Análisis de los resultados obtenidos en la evaluación de la métrica TOC

Durante la evaluación de la métrica TOC los resultados obtenidos demuestran que los atributos de calidad de las clases se encuentran en un nivel satisfactorio, de manera que se puede confirmar la elevada reutilización con un 57 % (elemento clave en el proceso de desarrollo de software) y cómo se reducen la responsabilidad y la complejidad de implementación en un 57 %.

Resultados obtenidos de la aplicación de la métrica RC y su influencia en los atributos de calidad

Como resultado de la evaluación de la métrica RC se obtuvo lo siguiente (Figura 30 y Figura **31**31):

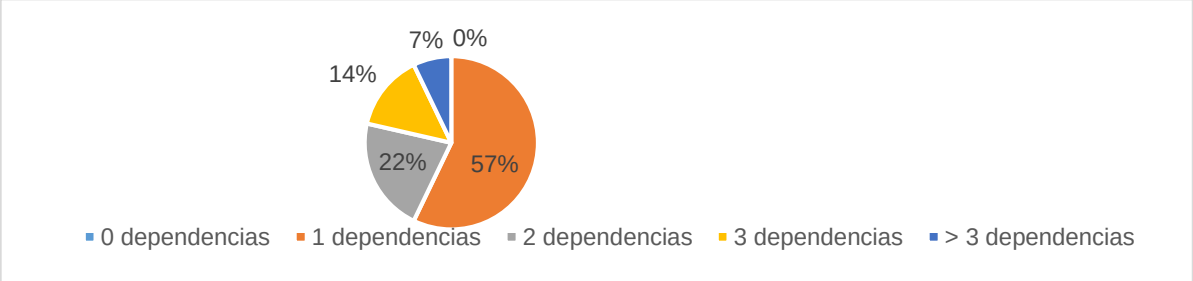


Figura 30: Resultados de la evaluación.

Representación en porciento de la incidencia de los resultados obtenidos en cada atributo de calidad:

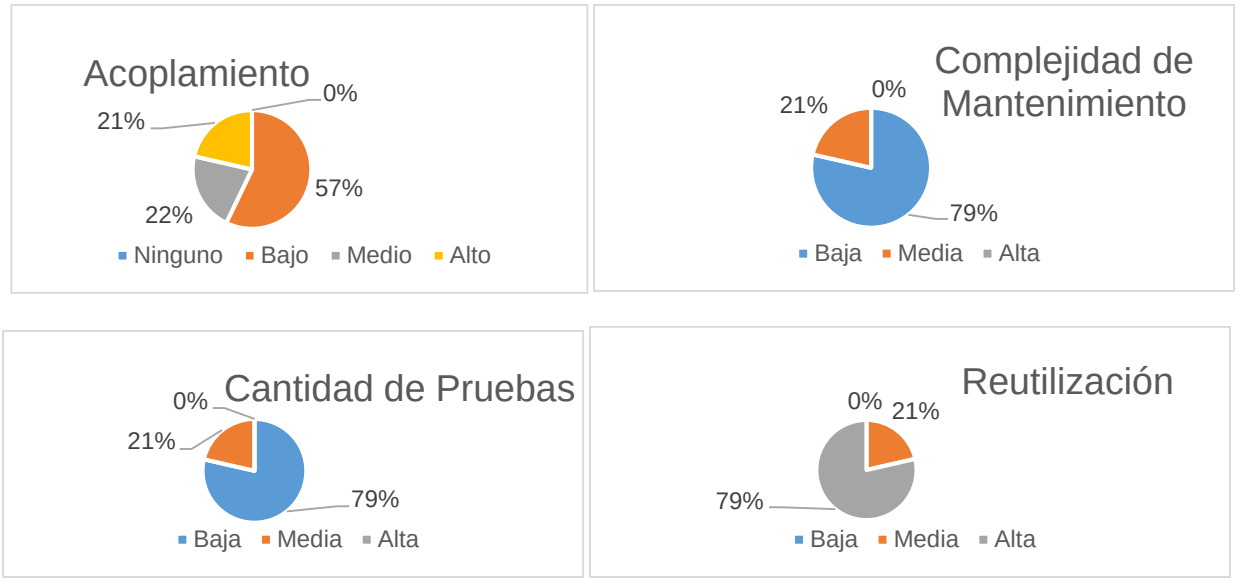


Figura 31: Representación en porciento de los resultados.

Análisis de los resultados obtenidos en la evaluación de la métrica RC

Luego de aplicarse la métrica de diseño RC y obtenidos los resultados de la evaluación del instrumento de medición de la métrica, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que las clases empleadas poseen menos de 3 dependencias de otras clases, lo que lleva a evaluaciones positivas de los atributos de calidad involucrados (acoplamiento bajo con un 57%, baja complejidad de mantenimiento 79%, baja cantidad de pruebas 79% y alta reutilización con un 79%). Favoreciendo de esta manera la reutilización de las clases así como la modificación e implantación del diseño.

Teniendo en cuenta el rango de evaluación y el análisis llevado a cabo se concluye que el sistema presenta un diseño con buena calidad.

2.5 Implementación

La implementación es el principal flujo de trabajo en la fase de construcción. En ella se describe cómo los elementos del modelo de diseño se implementan en función de componentes y, por ende, en piezas más

manejables por el programador. Tiene como objetivo llevar a cabo la implementación de cada una de las clases significativas del diseño (Pressman, 2005).

2.5.1 Diagrama de componentes

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos (Pressman, 2005). La Figura **32**32 muestra el diagrama de componentes utilizado en la investigación.

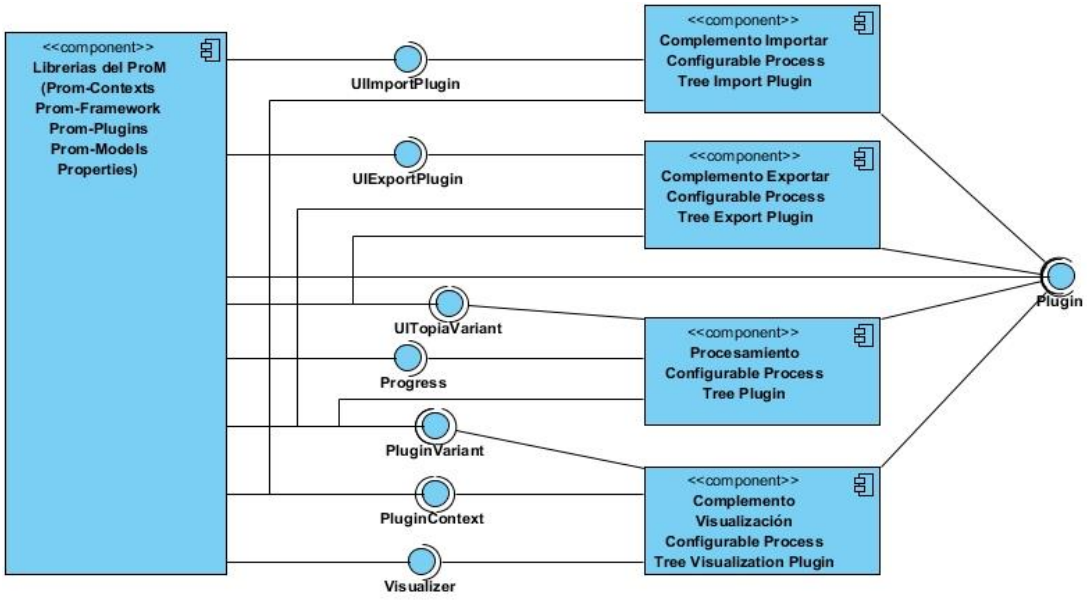


Figura 32: Diagrama de componentes.

2.5.2 Diagrama de despliegue

El diagrama de despliegue es un tipo de diagrama UML que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes (Pressman, 2005). En este caso el sistema puede funcionar perfectamente en una sola computadora, no necesita de ningún otro componente que se necesite para correr el software (Figura **33**33).

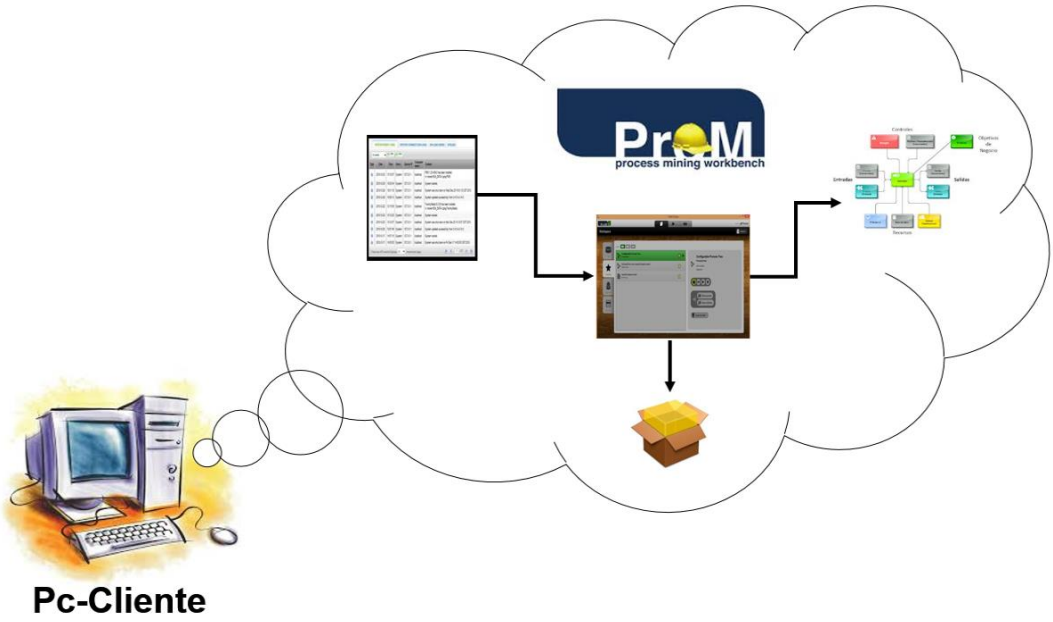


Figura 33: Diagrama de despliegue.

2.5.3 Estándares de codificación

Los estándares de codificación son aquellos que permiten entender de manera rápida y sencilla el código empleado en el desarrollo de un software. Garantizan el mantenimiento óptimo de dicho código por parte del programador. A continuación se muestran algunas pautas definidas por el equipo de desarrollo.

- Todas las nomenclaturas a utilizar se definirán en idioma español e inglés.
- Los identificadores para las variables y los parámetros serán con letras en minúsculas y en caso de ser un nombre compuesto se divide cada palabra con un guion bajo.
- En caso de nombrarse los métodos con una sola palabra, esta se escribe en minúsculas y en caso de ser un nombre compuesto, las palabras que lo conforman se escriben juntas, de la segunda en adelante se escriben con letra inicial mayúscula.

2.6 Conclusiones parciales

Se aplicó la metodología AUP generando los artefactos necesarios para orientar y describir el desarrollo de la solución. En el modelo de dominio se describieron los conceptos importantes y sus relaciones.

Se aplicaron algunas técnicas para capturar requisitos, como análisis de sistemas existentes, revisión de documentos y tormentas de ideas, que permitieron identificar y describir detalladamente nueve requisitos funcionales y determinar propiedades o cualidades que el sistema debe tener. Se estructuraron las clases en paquetes para organizarlas y comprenderlas mejor.

A través del diagrama de clases se presentó una vista general de las principales clases utilizadas en el diseño, sus atributos y métodos, así como la relación que existe entre ellas. Además, se aplicaron patrones GRASP y GoF que permitieron darle soluciones a problemas comunes en el desarrollo de software.

Para implementar el sistema el mismo fue dividido en cinco componentes relacionados por interfaces y se definieron los estándares de codificación a tener en cuenta. Se aplicaron las métricas TOC y RC para averiguar cuán bien están definidas las clases y el sistema, donde los resultados mostraron que el nivel de definición de las clases y sus relaciones fue satisfactorio y que el mismo presenta un diseño con buena calidad.

Capítulo 3: Validación de la solución propuesta

3.1 Introducción

En este capítulo se describen las pruebas de caja blanca y caja negra realizadas al complemento desarrollado para evaluar su calidad y comprobar sus funcionalidades en los diferentes escenarios, verificando en todos los casos que los resultados de las pruebas sean los esperados. Además, se realiza una consulta a especialistas para evaluar la solución.

3.2 Pruebas de Software

Las pruebas tienen gran importancia en el desarrollo de un software, ya que permiten detectar y corregir errores tempranamente y evaluar la calidad del producto que se desarrolló. Se realizan a través de diferentes fases mediante la aplicación de pruebas concretas para validar que las suposiciones hechas en el diseño y que los requerimientos se estén cumpliendo satisfactoriamente. (Jacobson, 2000).

Se aplican pruebas de caja blanca para examinar el programa en varios puntos sobre el código, para comprobar los métodos internos del subsistema y determinar si el estado real coincide con el esperado. Además, se aplican pruebas de caja negra para probar que las funcionalidades del subsistema son operativas.

3.2.1 Pruebas de Caja Blanca

La prueba de caja blanca es considerada como uno de los tipos de pruebas más importantes que se le aplican a los software, logrando como resultado que disminuya en un gran por ciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad (Pressman, 2005).

Existen dos formas de realizar las pruebas de caja blanca (PATTON, 2005):

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 y se observa el código mientras se ejecuta. Es decir, 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.

Se seleccionó la variante de pruebas de caja blanca dinámica. Para su realización se utilizó el marco de trabajo JUnit, ya que ofrece las funcionalidades necesarias para implementar pruebas en un proyecto desarrollado en Java, brinda un conjunto de bibliotecas que se integran fácilmente al IDE seleccionado (Eclipse), permite la automatización de las pruebas de aplicaciones en Java, tiene un conjunto de clases que el desarrollador puede utilizar para construir sus casos de prueba y ejecutarlos automáticamente. Además, mejora el diseño de implementación, haciéndolo flexible y probable (TAHCHIEV, y otros, 2010) y cuenta con una interfaz simple que informa si cada una de las pruebas realizadas o conjunto de pruebas falló, pasó o fue ignorada.

Para realizar las pruebas se siguieron los siguientes pasos (TAHCHIEV, y otros, 2010):

- Crear un caso de prueba por cada clase implementada.
- Configurar, en el caso de prueba, los ficheros que permiten comunicar las clases de las capas de presentación y lógica de negocio.
- Definir los métodos a probar dentro de cada caso de prueba, incluyendo los parámetros de entrada.
- Realizar pruebas a los métodos.

En la Figura 344 se evidencia cómo se le realizaron las pruebas de caja blanca a la clase controladora **ConfigurarProcessTreeManager**, localizada en el paquete **plugin**, demostrándose mediante la aplicación de la prueba al método **convertSubprocessToProcessTree** ().

```
46  /**
47   * * Convierte un Subprocess a un ProcessTree
48   *
49   * @return ProcessTree
50   */
51  public ProcessTree convertSubprocessToProcessTree() {
52
53      ProcessTree tree = new ProcessTreeImpl();
54
55      Block root = new AbstractBlock.Xor("ROOT");
56      root.setProcessTree(tree);
57      tree.addNode(root);
58      tree.setRoot(root);
59
60      BuildTree(this.subprocess, tree, root);
61
62      return tree;
63  }
```

Figura 34: Prueba de caja blanca aplicada al método convertSubprocessToProcessTree ().

Para realizar la prueba a dicho método primeramente se creó una clase (Figura 355) que heredara de la clase TestCase de JUnit. En este caso la clase se nombra **ConfigurarProcessTreeManagerTest** y en ella se declaran los objetos necesarios para probar los métodos de las clases en el método setUp ().

ConfigurableProcessTreePlugin ▸ src ▸ org.processmining.cfgprocesstree.test

```
27 public class ConfigurableProcessTreeManagerTest {
28     private ProcessTree treeEsperado;
29     public void setUp() throws Exception {
30         this.treeEsperado = new ProcessTreeImpl();
31         Edge edge;
32         // Xor( Seq( F ,  Z ) , B )
33         Block xor = new AbstractBlock.And("Xor");
34         xor.setProcessTree(this.treeEsperado);
35         this.treeEsperado.addNode(xor);
36         this.treeEsperado.setRoot(xor);
37         Block seq = new AbstractBlock.Seq("Seq");
38         seq.setProcessTree(treeEsperado);
39         this.treeEsperado.addNode(seq);
40         edge = xor.addChild(seq);
41         seq.addIncomingEdge(edge);
42         this.treeEsperado.addEdge(edge);
43         Task f = new AbstractTask.Manual("F");
44         f.setProcessTree(this.treeEsperado);
45         this.treeEsperado.addNode(f);
46         edge = seq.addChild(f);
47         f.addIncomingEdge(edge);
48         this.treeEsperado.addEdge(edge);
49         Task tau = new AbstractTask.Manual("Z");
50         tau.setProcessTree(this.treeEsperado);
51         this.treeEsperado.addNode(tau);
52         edge = seq.addChild(tau);
53         tau.addIncomingEdge(edge);
54         this.treeEsperado.addEdge(edge);
55         Task a = new AbstractTask.Manual("B");
56         a.setProcessTree(treeEsperado);
57         this.treeEsperado.addNode(a);
58     }
```

Figura 35: Clase ConfigurarProcessTreeManagerTest ().

En la Figura 366 se muestra un ejemplo de cómo se prueba un método.

En este caso se prueba que el método **convertSuprocessToProcessTree ()** no devuelve un dato nulo.

```
74  /**
75   * Probando que el ConvertSubprocessToProcessTree
76   * que el método convertSubprocessToProcessTree no de resultado Null
77   * Test method for {@link org.processmining.cfgprocesstree.plugin.ConfigurableProcessTreeManager#convertSubprocessToProcessTree()}.
78   */
79  @Test
80  public void testConvertSubprocessToProcessTree() {
81      Subprocess subprocess = new Subprocess();
82      ConfigurableProcessTreeManager manager = new ConfigurableProcessTreeManager(subprocess);
83      ProcessTree tree = manager.convertSubprocessToProcessTree();
84      assertNotNull(tree);
85  }
```

Figura 36: Ejemplo de prueba del método convertSuprocessToProcessTree ().

El resultado de la prueba realizada a todos los métodos de la clase **ConfigurarProcessTreeManager** fue satisfactorio, evidenciándose por el color verde mostrado en la barra de la parte superior izquierda (en caso de existir fallos la barra tendría color rojo), como se muestra en la Figura 377. En dicho escenario se ejecutaron 33 casos de pruebas, donde el 100 % tuvo 0 errores (errors del inglés) y 0 fallos (failures del inglés).

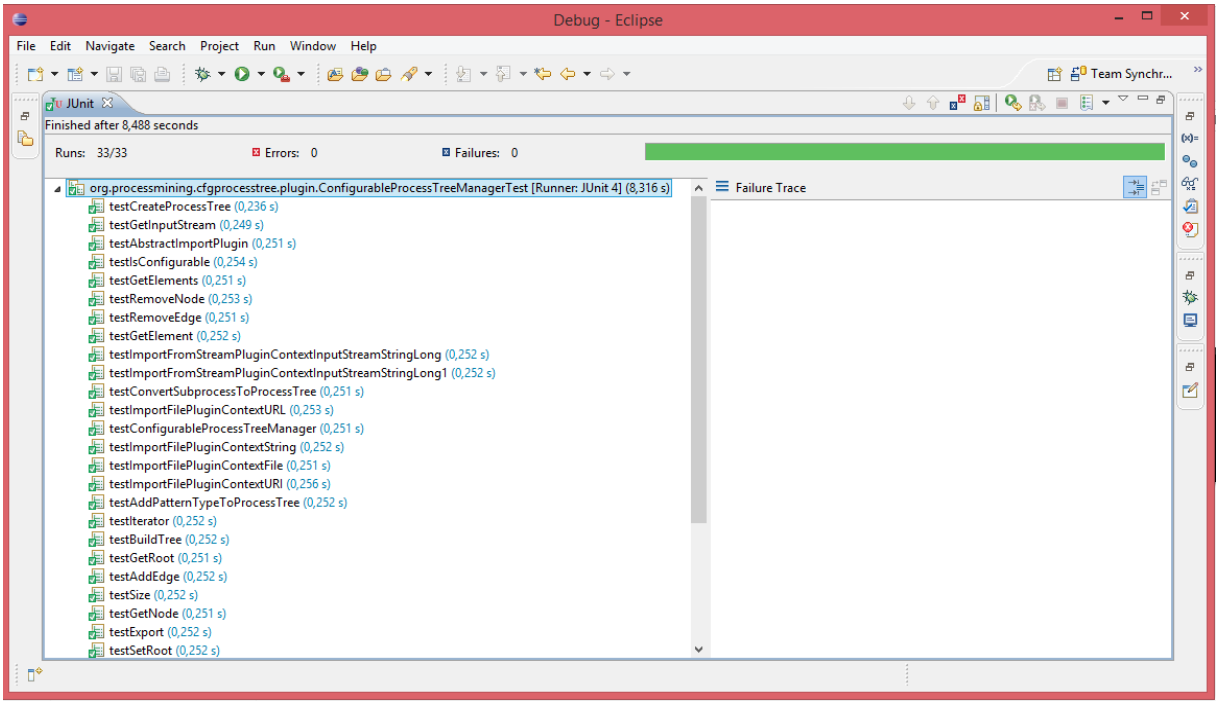


Figura 37: Resultados de la pruebas realizadas.

Es importante aclarar que se utilizó JUnit para evaluar si el funcionamiento de cada método de las clases se comportaba como se esperaba, lo que permitió confirmar que los errores detectados durante la implementación habían sido corregidos por los desarrolladores.

3.2.2 Pruebas de Caja Negra

Las pruebas de caja negra verifican las especificaciones funcionales sin tener en cuenta la estructura interna del programa y son realizadas sin el conocimiento interno del producto. Con este tipo de pruebas se intenta encontrar funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a las bases de datos externas, errores de rendimiento y que las entradas se aceptan de forma adecuada y se produce un resultado correcto. Se pretende demostrar que la integridad de la información externa se mantiene, saber qué es lo que hace el software pero sin entrar en detalles de código, es decir, qué hace y no cómo lo hace. Por ello se realizan sobre la interfaz del sistema controlando los datos de entrada y de salida (Pressman, 2005).

Existen varias técnicas para desarrollar la prueba de caja negra, entre ellas están:

- Partición de Equivalencia: Divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. En esencia esta técnica 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.
- Análisis de Valores Límites: Los errores tienden a darse más en los límites del campo de entrada que en el centro. Esta técnica lleva a una elección de casos de prueba que ejerciten los valores límites.
- Prueba basada en grafos: Permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Para la realización de las pruebas se utilizó la técnica Partición Equivalente. Esta es una de las técnicas más efectiva ya que permite comprobar los valores válidos e inválidos de todas las entradas existentes en el software, además de que permite obtener un conjunto de pruebas que reducen el número de casos de prueba que se deben realizar para evaluar correctamente el software.

Se realizaron dos iteraciones y un caso de prueba para cada requisito (ver *Pruebas del software*). En la primera iteración se detectaron 12 no conformidades, de las cuales 9 fueron de la aplicación y 3 de

documentación. Dentro de las no conformidades referentes a la aplicación se encontraron 7 de validación y 2 de interfaz. En cuanto a las de documentación se encontraron 2 no conformidades de redacción y 1 de ortografía.

Todas estas no conformidades fueron solucionadas seguidamente de haberlas detectado, lo que permitió que el complemento pasara a una segunda iteración en la que no se detectaron no conformidades, obteniendo excelentes resultados.

3.3 Validación de la solución

Para validar la solución se le realizó una consulta a cuatro especialistas en el tema, los cuales mediante un cuestionario que recoge los principales puntos de la investigación, brindaron sus criterios y evaluaron el complemento desarrollado. A continuación se muestra la relación de estos especialistas (ver Tabla 5).

Tabla 5: Relación de especialistas consultados.

Especialistas	**Categoría docente**	**Categoría científica**	**Nivel Profesional**	**Años de experiencia**
Esp.1	Asistente	Doctor	Doctor	7 años
Esp.2	Instructor	Sin categoría	Ingeniero	6 años
Esp.3	Asistente	Sin categoría	Ingeniero	5 años
Esp.4	Sin categoría	Sin categoría	Ingeniero	1 año

Las preguntas realizadas (*ver Anexo # 1*) arrojaron como resultado lo siguiente:

- Se visualizan todas las variantes de forma simultánea en el Árbol de Procesos Configurable.
- Se puede exportar el Árbol de Procesos Configurable.
- Se le pueden aplicar a los nodos del Árbol de Procesos Configurable las opciones de configuración como bloquear, ocultar y degradar.
- Se pueden reubicar los nodos dentro del área de trabajo.

- Se puede generar un Árbol de Procesos Configurable a partir de una configuración contenida en un Árbol de procesos.
- Se puede deshacer la última configuración del Árbol de Procesos Configurable.
- Lo anterior permitió confirmar que el Árbol de Procesos Configurable ofrece una visión más general del proceso que el Árbol de Variantes.

Por todos los elementos antes expuestos y después de realizar las preguntas pertinentes y obtener resultados satisfactorios se llega a la conclusión de que el complemento cumple con el objetivo trazado en la investigación.

3.4 Conclusiones del capítulo

En este capítulo se realizaron pruebas dinámicas de caja blanca para examinar el código del programa en varios puntos y comprobar los métodos internos del subsistema. Para ello se empleó el marco de trabajo JUnit aprovechando las funcionalidades y bibliotecas que ofrece. Los resultados mostraron que no existían errores en el código.

Además se realizaron pruebas de caja negra utilizando la técnica Partición Equivalente para probar que las funcionalidades del subsistema eran operativas, donde se verificó en todos los casos después de una segunda iteración que no existían no conformidades, es decir, que las funcionalidades del sistema se encuentran acorde a las necesidades definidas en los requisitos.

Además, se realizó una consulta a especialistas en el tema que permitió confirmar la validez de la solución propuesta y el cumplimiento del objetivo de la investigación.

Conclusiones

La fundamentación teórica realizada posibilitó detectar las limitantes de la técnica Minería de Variantes y definirla como punto de partida de la investigación.

El estudio realizado permitió, además, seleccionar la notación Árbol de Procesos para modelar los procesos configurables y utilizar la técnica Árbol de Procesos Configurable para solventar las limitantes de la técnica Árbol de Variantes.

Se aplicó la metodología AUP para guiar el proceso de desarrollo del sistema, generando artefactos que permitieron orientar y documentar el desarrollo de la solución.

Se aplicaron métricas para evaluar el diseño donde se concluyó que el sistema presenta un diseño con buena calidad. Además se realizaron pruebas de caja negra y caja blanca para evaluar el sistema implementado y se realizó una consulta a especialistas en el tema para validar la solución en donde se obtuvieron excelentes resultados, corroborando así el cumplimiento del objetivo de la investigación y logrando una solución de alta calidad.

Finalmente se obtuvo un sistema multiplataforma y totalmente libre, amigable, sencillo, de fácil manipulación por los usuarios familiarizados con el ProM, que permite mostrar todas las variantes de un proceso y configurarlas.

Recomendaciones

1. Implementar el uso de colores para diferenciar los patrones de control de flujo de las actividades.
2. Brindar la opción de exportar el Árbol de Procesos Configurable como una imagen (.jpg).
3. Mejorar la interacción con el usuario, usando mensajes de información o leyendas.

Bibliografía.

Aalst, W. M. P. v. d, Adriansyah, A., van Dongen, B. 2012. *Replaying History on Process Models for Conformance Checking and Performance Analysis.* s.l. : WIREs Data Mining and Knowledge Discovery 2(2), 2012. págs. 182–192.

Aalst, W. M. P. v. d, Medeiros, A. K. A., Bose, R. P. J. C., Arcieri, F., Rozinat, A.Seguel, H. P., et al. 2011. *Manifiesto sobre Minería de Procesos.* versión final. 2011.

Aalst, W.M.P.v.d. 2011. *Process Mining.Discovery,Conformance and Enhancement of Business Processes.* London, New York: Springer Heidelberg Dordrecht : s.n., 2011.

Aalst, W.M.P.v.d,Reijers,A.,& van Dongen,A.K. 2007. *Business Process Mining: An Industrial Application. Information Systems.* 2007. págs. 713-732.

Aalst, W.M.P.v.d,Rubin,V.,Verbeek, H. M. W.,van Dongen,B.F.,Kindler,E.,& Günther,C.W. 2010. *Process Mining: A Two-Step Approach using Transition Systems and Regions. Software and Systems Modeling.* 2010. págs. 87-111. Vol. 9(1).

Aalst, W.M.P.v.d,Weijters, A.J.M.M.,et al. 2004. *Workflow Mining: Discovering Process Models from Event Logs.* 2004. págs. 1128–1142. IEEE Transactions on Knowledge and Data Engineering.

ALUR, D. y CRUPI, J. 2003. *Core J2EE patterns: best practices and design strategies. .* s.l. : Prentice Hall PTR, , 2003. p. 0131422464.

Becker, J., Delfmann, P., Dreiling, A., Knackstedt, R., Kuropka, D. 2004. *Configurative Process Modeling–Outlining an Approachto increased Business Process Model Usability.* Proceedings of the 15th IRMA International Conference : s.n., 2004.

BOSE R.P.J.C. 2012. *Process Mining in the Large: Preprocessing, Discovery, and Diagnostics.* Eindhoven University of Thechnology. : s.n., 2012.

Bose, R.P.J.C. & Aalst,W.M.P.v.d. 2012. *Process Diagnostics using Trace Alignment: Opportunities, Issues, and Challenges. Information Systems.* 2012. págs. 117-141.

—. 2010. *Trace Alignment in Process Mining: Opportunities for Process Diagnostics.* 2010. págs. 227-242. International Conference on Business Process Management. ISBN:3-642-15617-7.

BPM. 2014. Acerca de BPM. *AmericaVeintiuno.* [En línea] AmericaVeintiuno, 2014. [Citado el: 4 de junio de 2014.] http://www.americaveintiuno.cl/?page_id=1823.

Brache, A. P. 2010. *What is a Process? Why should you care?* s.l. : Electronic Version, 2010.

Buijs, J.C.A.M.,B.F.van Dongen,and Aalst,W.M.P.v.d. 2013. *Mining Configurable Process Models from Collections of Event Logs.* 2013.

Buijs, J.C.A.M.,van Dongen,B.F.,van der Aalst, W.M.P. 2012. *On the Role of Fitness, Precision, Generalization and Simplicity in Process Discovery.* Springer, Heidelberg : s.n., 2012. págs. 305–322. Vol. vol.7565.

Davila, Dr.Agustin Lage. 2004. *LA ECONOMIA DEL CONOCIMIENTO Y EL SOCIALISMO.* 2004.

DIXON, M. 2004 . *A single CASE environment for teaching and learning.* Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education. Software Engineering Institute. Leeds, UK, : s.n., 2004 . Proceedings of the 9th annual SIGCSE conference on Innovatio.

Dongen, B.F.,Adriansyah,A. 2010. *Process Mining: Fuzzy Clustering and Performance Visualization.* Springer Berlin Heidelberg : In Business Process Management Workshops S. Rinderle-Ma,Sadiq, and F. Leymann, Editors, 2010. págs. 158--169.

E. P. Hernández, R. L. Jiménez, D. Pérez Alfonso, R. Yzquierdo Herrera. 2014. *Variantes del proceso: un enfoque para el diagnóstico de procesos de negocio.* La Habana Cuba : s.n., 2014. IX Peña Tecnológica. 2014.

EMIT. 2004. *a process mining tool.* 2004.

Figueroa, Roberth G. y Solís, Camilo J. 2011. *Metodologías Tradicionales vs. Metodologías Ágiles.* 2011.

Fornaris, Maite Sánchez y Rabí, Dayana Alcantara. 2010. *Propuesta de una guía de métricas ara evaluar el desarrollo de los Sistemas de Información.* 2010.

Goedertier, S.,et al. 2008. *Process Mining as First-Order Classification Learning on Logs with Negative Events.* s.l. : International Workshops, 2008. págs. 42–53. Lecture Notes in Computer Science.

—. 2009. *Robust Process Discovery with Artificial Negative Events.* 2009. págs. 1305-1340. Vol. 10, Journal of Machine Learning Research. 1532-4435.

GÓMEZ, O. T. y P. P. R. LÓPEZ. 2010. *Criterios de selección de metodologías de desarrollo de software Ind.* 2010.

Gottschalk, F.,Aalst, W.M.P.v.d,Jansen-Vullers,M.H.,La Rosa,M. 2008. *Configurable Workflow Models.* International Journal of Cooperative Information Systems : (IJCIS) 17(2), 2008. International Journal of Cooperative Information Systems.

Gottschalk, F.,Aalst,W.M.P.v.d,Jansen-Vullers,M.H. 2008. *Mining Reference Process Models and their Configurations.* [ed.] R.,Tari, Z.,Herrero,P. Meersman. Springer, Heidelberg : s.n., 2008. págs. 263–272. Vol. vol.5333.

Gottschalk, F.,van der Aalst,W.M.P.,Jansen-Vullers, M.H. 2008. *Merging Event-driven Process Chains.* [ed.] R.,Tari,Z. Meersman. Springer, Heidelberg : s.n., 2008. págs. 418–426. Vol. vol.5331.

Günther, C.W. 2009. *Process Mining in Flexible Environments.* 2009. Unpublished doctoral thesis, Eindhoven University of Technology.

Günther, C.W.,& Aalst,W.M.P.v.d. 2007. *Fuzzy Mining: Adaptive Process Simplification Based on Multi-Perspective Metrics.* 2007. Paper presented at the International Conference on Business Process Management.

Harrington, H. J. 2005. *Mejoramiento de los procesos de la empresa.* Santa Fe de Bogotá : McGraw_ Hill Co, 2005.

J. L. Cordero. 2014. *Metodologías Ágiles " Proceso Unificado Ágil (AUP)".* La Paz, EL Alto : s.n., 2014.

Jacobson, I., G. Booch, and J. Rumbaugh. 2000. *El Proceso Unificado de Desarrollo de Software.* 2000.

Juran, J.M. 2000. *Juran's Quality Handbook.* s.l. : McGraw-Hill Publishing Co, 2000.

Kidd, Mark Lorenz y Jeff. 1994. *Object-Oriented Software Metrics.* New Jersey,Englewood Cliffs : s.n., 1994.

La Rosa M., WIL M.P. VAN DER AALST., MARLON DUMAS, FREDRIK P. MILANI. 2013. *Business Process Variability Modeling: A Survey.* 2013.

La Rosa, M.,Dumas,M.,Uba,R.,Dijkman,R. 2012. *Business Process Model Merging: An Approach to Business Process Consolidation.* 2012. ACM Transactions on Software Engineering and Methodology.

Larman, Graig. 1999. *UML y Patrones. Introducción al análisis y diseño orientado a objetos y al proceso unificado.* Mexico : PRENTICHE HAL, 1999.

Li, C.,Reichert,M.,Wombacher,A. 2010. *The MINADEPT Clustering Approach for Discovering Reference Process Models Out of Process Variants.* 2010. págs. 159–203. International Journal of Cooperative Information Systems.

Madrid, Universidad Politécnica de. 2005. *Patrones del "Gang of Four".* Madrid : s.n., 2005.

Medeiros, A.K.A.d. 2006. *Genetic Process Mining.* 2006. Unpublished PhD.thesis,Technische Universiteit Eindhoven.

Medeiros, A.K.A.d.,et al. 2003. *Workflow Mining: Current Status and Future Directions.* s.l. : in The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE, 2003. págs. 389–406. Vol. vol.2888, Lecture Notes in Computer Science.

NC/ISO. 2005. *"Sistemas de Gestión de la Calidad — Fundamentos y Vocabularios".* 2005.

PARADIGM, VISUAL. 2012. *Visual Paradigm 6.4.* 2012.

Patrones GRASP. 2011. Patrones GRASP. *Patrones GRASP.* [En línea] 10 de Noviembre de 2011. [Citado el: 10 de Noviembre de 2011.] http://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.

PATTON, R. 2005. *Software testing,.* Sams Publishing, : s.n., 2005.

Pressman, R. 1999. *Software Engineering.* A Practitioner's Approach. USA : s.n., 1999.

Pressman, Roger S. 2005. *Ingeniería del Software: Un enfoque práctico. 2005.* 2005.

Roya ZarehFarkhady, Seyyed Hasan Aali. 2012. *A Two Phase Approach for Process Mining in Incomplete and .* 2012. págs. 415-420. Vol. vol.2188, International MultiConference of Engineers and Computer Scientists.

Rozinat, A.,& Aalst,W.M.P.v.d. 2008. *Conformance Checking of Processes Based on Monitoring Real Behavior.* 2008.

Rummler, G.A.,& Brache,A.P. 2006. *Improving Performance: How do manage the White Space on the Organization Chart.* 2nd ed. San Francisco : s.n., 2006. Jossey-Bass Inc.

Schunselaar, D.M.M.,Verbeek,E.,Aalst,W.M.P.v.d,Raijers,H.A. 2012. *Creating Sound And Reversible Configurable Process Models Using CoSeNets.* [ed.] W., Kriksciuniene, D., Sakalauskas, V. Abramowicz. Springer, Heidelberg : Springer, Heidelberg, 2012. págs. 24–35. Vol. vol.117.

Song, M.,& Aalst,W.M.P.v.d. 2007. *Supporting Process Mining by Showing Events at a Glance.* Montreal : s.n., 2007. Paper presented at the 17th Annual Workshop on Information Technologies and Systems.

Strnadl, C.F. 2005. *Aligning Business and IT: The Process-Driven Architecture Model.* [ed.] IEEEXplore. s.l. : 2005-EUROCON, 2005. Vol. Vol.2, The International Conference.

TAHCHIEV, P. y F. LEME. 2010. *JUnit in action. .* s.l. : Manning Publications Co., 2010. p. 1935182021].

Tiwari, A.,et al. 2008. *A review of business process mining: state -of-the-art and future trends.* 2008. págs. 5-22. Vol. vol. 14. ISSN: 1463-7154. DOI: 10.1108/14637150810849373.

Vázquez, Cubos D E. 2007. *La importancia de los procesos de negocio en las IT.* 2007.

Weijters, A.J.M.M. & Aalst,W.M.P.v.d. 2003. *Rediscovering Workflow Models from Event-Based Data using Little Thumb.* 2003. págs. 151-162. Vol. vol.10. ISSN: 1069-2509.

WEIJTERS, A.J.M.M. and J.T.S. 2010. *RIBEIRO: Flexible Heuristics Miner (FHM).* Eindhoven : Working Paper Series, WP 334, 2010. Eindhoven University of Technology.

Yzquierdo-Herrera, R. 2012. *Modelo para la estimación de información ausente en las trazas usadas en la Minería de Procesos.* 2012.

Yzquierdo-Herrera, R.,Silverio-Castro R.,Lazo-Cortés M. 2012. *Sub-process discovery: Opportunities for Process Diagnostics.* Habana : s.n., 2012.

Anexos

Anexo 1

Preguntas:

Se visualizan todas las variantes de forma simultánea en el árbol de procesos configurable

Exportar el árbol de procesos configurable

Permite aplicar opciones de configuración teniendo en cuenta:

- Permite ocultar una parte del proceso
- Permite Bloquear una parte del proceso
- Permite degradar nodos operadores

Se visualizar un árbol de proceso configurable.

Se pueden reubicar los nodos dentro del área de trabajo

Visualizar un árbol de proceso configurable a partir de una nueva configuración del árbol de proceso.

Se puede deshacer la última configuración del árbol de procesos configurable.

Anexo 2

AVAL DE ACEPTACIÓN

Ing. Dina Yaksilik Torres Sakipova declara:

Ante mí se ha presentado el proyecto de investigación y desarrollo "Complemento para generar un árbol de procesos configurable", de los autores Carlos Llama Rodríguez y Daniel Pulido Dorta.

El producto constituye un valioso aporte a la investigación sobre minería de procesos que lleva a cabo el grupo de investigación de esta área, perteneciente a la Facultad 3 de la Universidad de las Ciencias Informáticas. Forma parte del trabajo investigativo de varias tesis de maestría de profesores que integran el grupo. El complemento muestra todas las variantes de un proceso y permite su configuración, de manera que se corresponde con lo solicitado por el cliente.

De esta forma se expresa la satisfacción respecto al resultado investigativo y se reconoce el esfuerzo de los estudiantes durante el desarrollo de la Tesis.

Para que así conste, se expide el presente Aval de Aceptación.

Dado en La Habana a los **3** días del mes de **junio** del **2014**. "Año 56 de la Revolución".

Por: Ing. Dina Yaksilik Torres Sakipova, Miembro del Grupo de Investigación de Minería de Procesos.

Facultad 3

Universidad de las Ciencias Informáticas

La Habana, Cuba