UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 3

# Algoritmo para construir variantes de proceso a partir de registros de eventos con alto número de sucesiones indirectas entre las tareas.

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

**Autor:**
**Raidel González Rojas**

**Tutor:**
**DrC. Damián Pérez Alfonso**

**La Habana, julio de 2016**
**"Año 58 de la Revolución"**

## DECLARACIÓN DE AUTORÍA

Declaro ser el autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

| | |
|---|---|
| _____ | _____ |
| Raidel González Rojas | Damián Pérez Alfonso |
| Autor | Tutor |

## DEDICATORIA

*A mis padres, por todo su sacrificio.*

*A Yanet y mis primos, por motivarme a ser un ejemplo a seguir.*

*A mis tíos Vladimir, Norma y los payos, por enseñarme que familia va más allá de la sangre.*

*A mis abuelos, que sus canas sigan siendo sinónimo de sabiduría.*

*A mis tíos y tías, por todo su apoyo.*

*A mi abuelo Rigo, que ya no está entre nosotros.*

*Al resto de mi inmensa familia que, aunque no caben aquí siempre los tendré presentes.*

## AGRADECIMIENTOS

*A mi tutor Damián, por mostrarme el mundo de la minería de procesos y por todas las lecciones de la vida.*

*A mi familia, por todo el apoyo y por creer en mí.*

*A los profesores Yusniel y Maybel por creer en mí y por todo el apoyo.*

*A los ingenieros González Leyva, Suárez Mastrapa, Sirés González, Molina Suárez, Junquera Falcón y a Liorge por ayudarme con su ya creciente experiencia como profesionales.*

*Al resto de mis amigos por su preocupación*

*A los profesores del departamento de Programación, por todo el apoyo y ayuda.*

*A todos los maestros que me han formado a lo largo de mi vida. Por brindarme su recurso más valioso, el tiempo.*

*A todas las personas que han tendido su mano desinteresada y me han ayudado durante este período y que desafortunadamente no caben aquí.*

***RESUMEN***

Los Sistemas de Gestión de Procesos de Negocios permiten almacenar registros de los eventos de los procesos que gestionan. La minería de procesos impulsa el desarrollo de técnicas y herramientas que permiten extraer y analizar la información almacenada en los registros de eventos. La presencia de ruido y ausencia de información en los registros de eventos genera una diversidad de comportamiento que dificulta la identificación de patrones de control de flujo por parte de los algoritmos de descubrimiento. La Minería de Variantes es una técnica de descubrimiento para la obtención de variantes de procesos de negocio. El comportamiento que más afecta a la Minería de Variantes son las sucesiones indirectas, aumentando considerablemente su complejidad temporal. Los algoritmos de descubrimiento utilizan diferentes enfoques para tratar el comportamiento generado por el ruido y la ausencia de información. En esta investigación se propone un algoritmo que permite descubrir variantes de proceso en un menor tiempo independientemente del número de sucesiones indirectas. El algoritmo utiliza el enfoque de procesar las sucesiones indirectas en función de su frecuencia de aparición. Se muestra un análisis de los resultados de la aplicación del algoritmo ante registros de eventos que generan diferentes cantidades de sucesiones indirectas. Estos resultados son comparados con los obtenidos a partir de la Minería de Variantes.

**Palabras claves**: descubrimiento de procesos, minería de proceso, Minería de Variantes, sucesiones indirectas, variantes de modelos de proceso.

***ABSTRACT***

Business Process Management Systems allow to store event logs related to processes executions. Process mining boosts the development of tools and techniques that allow to extract and analyze the information stored in the event logs. The presence of noise and absence of information in the event logs generates a diversity of behavior that hampers the identification of control flow patterns by discovery algorithms. Variants Mining is a discovery technique to obtain business process variants. The behavior that most affects Variants Mining are indirect successions, greatly increasing their time complexity. Discovery algorithms use different approaches to deal with the behavior generated by noise and absence of information. An algorithm to discover process variants in a shorter time regardless of the number of indirect successions is proposed. The algorithm uses the approach of processing the indirect successions by their occurrence frequency. An analysis of the algorithm results on event logs with different amounts of indirect successions is presented. These results are compared with those obtained by the Variants Mining algorithm.

**Key words:** indirect successions, process discovery, process mining, process models variants, variants mining.

## ÍNDICE

## ÍNDICE DE TABLAS

## ÍNDICE DE FIGURAS

## INTRODUCCIÓN

En el desarrollo de las organizaciones la gestión de sus procesos ha tomado cada vez mayor protagonismo. El enfoque basado en procesos, base para la gestión de procesos, ha permitido una identificación y gestión sistemática de los procesos asociados a una organización, así como sus interacciones. Las definiciones de Mathias Weske realizan una combinación armoniosa de los procesos de negocio y la gestión de estos:

**Definición 1.** *Un proceso de negocio es una colección de actividades que son realizadas coordinadamente en un ambiente técnico y organizacional. La conjunción de estas actividades logra un objetivo del negocio. Cada proceso de negocio es establecido por una organización, pero con él pueden interactuar procesos de negocios de otras organizaciones* (Weske y Heidelberg 2007).

**Definición 2.** *La gestión de procesos de negocio incluye conceptos, métodos y técnicas para el diseño, administración, configuración, ajuste y análisis de los procesos de negocio* (Weske y Heidelberg 2007).

Como soporte a esta forma de organizar las actividades han surgido los Sistemas de Gestión de Procesos de Negocios (BPMS, por sus siglas en inglés). Gran parte de estos sistemas brindan la posibilidad de registrar trazas de las ejecuciones de los procesos que gestionan. Estos registros de ejecuciones de procesos, denominados registros de eventos, recopilan información sobre los estados que transcurren durante la ejecución de estos procesos. Con el fin de estudiar y aprovechar esta información surge la minería de procesos, disciplina científica que desarrolla técnicas y herramientas que permiten analizar los registros de eventos, extraer información a partir de ellos y presentar de forma explícita el conocimiento que contienen (van der Aalst 2011). La minería de procesos tiene como fin descubrir, monitorear y mejorar los procesos reales; para cada uno de ellos se corresponde una de las tres grandes tareas de la minería de procesos: el descubrimiento de procesos, la verificación de conformidad y el mejoramiento de modelos respectivamente. En la Figura 1 se puede observar la minería de procesos, sus etapas y cómo estas se utilizan en la gestión de procesos de negocio (Verbeek et al. 2011).

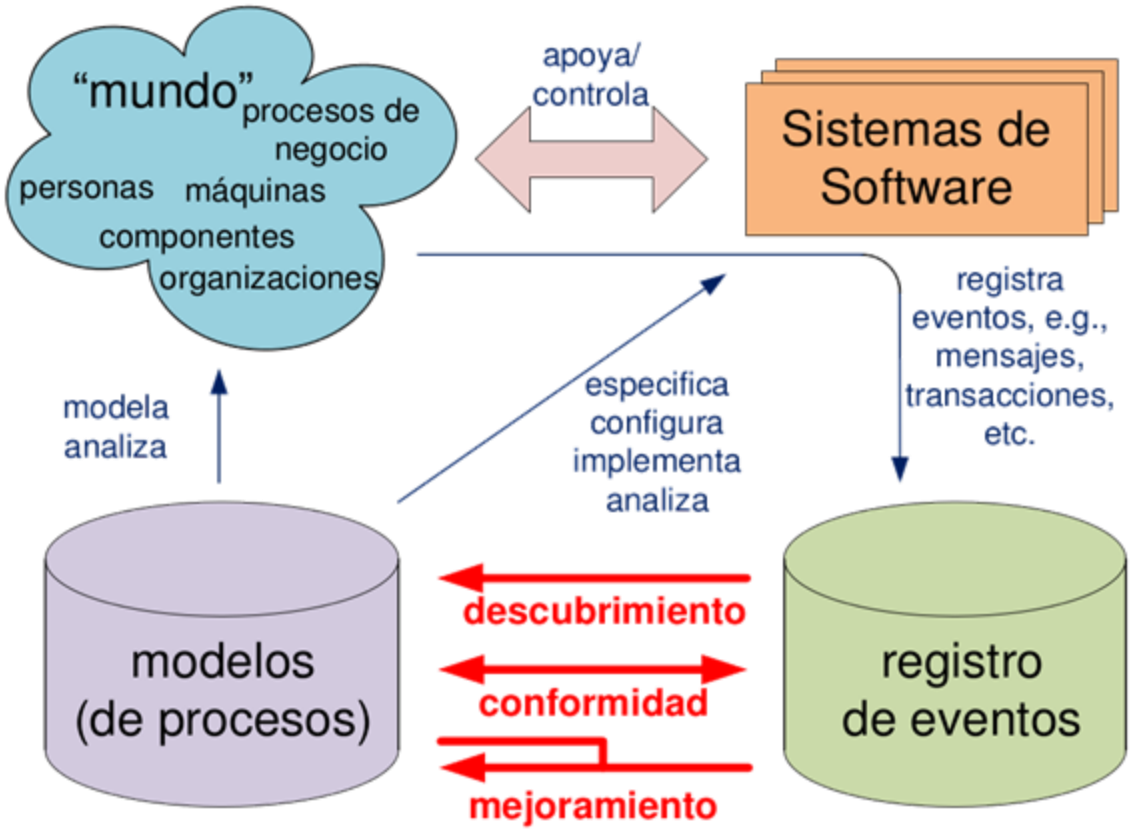


***Figura 1:*** *Posicionamiento de los tres tipos principales de minería de procesos: descubrimiento de procesos, chequeo de conformidad y mejoramiento de modelos* (van der Aalst 2011).

El ciclo de vida de la Gestión de Procesos de Negocio abarca las siete fases de un proceso de negocio. El ciclo comienza con la fase de (re)diseño donde se crea un nuevo modelo de proceso o se adapta uno existente. Junto a esta fase debe realizarse el análisis del modelo candidato y sus alternativas. Una vez listo el modelo del proceso este puede pasar a la fase de implementación o (re)configuración en un sistema de información para su ejecución. Durante la ejecución del proceso se podrán realizar pequeños ajustes sin rediseñarlo (fase de ajuste). Para cerrar el ciclo se analiza la ejecución del proceso en la fase de diagnóstico, lo cual puede iniciar una nueva fase de (re)diseño del proceso. En la Figura 2 se muestra con más detalles el ciclo de vida de la Gestión de Procesos de Negocio.

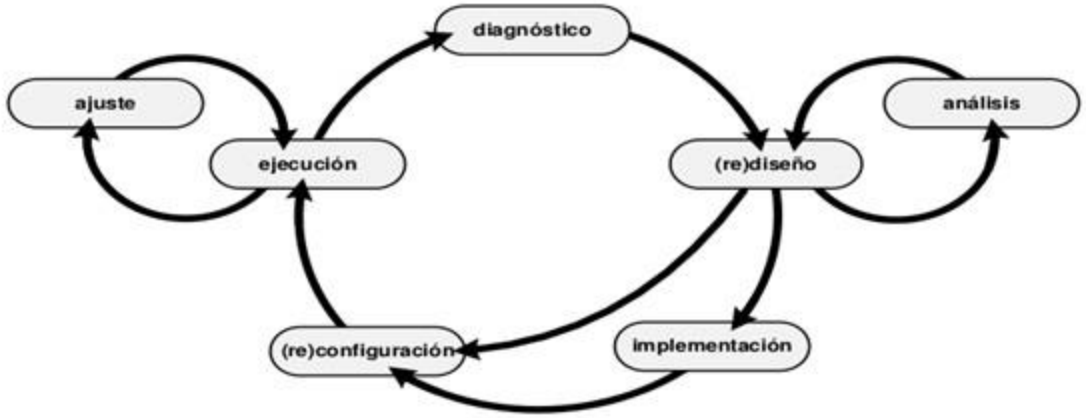


***Figura 2:*** *Ciclo de vida de BPM* (van der Aalst 2011).

En la fase de diagnóstico de proceso se evalúan y mejoran los modelos de procesos y sus implementaciones. Esta etapa comprende el chequeo de conformidad, la auditoría, el análisis de rendimiento, la detección de anomalías y la identificación de patrones comunes (Bose 2012). En la fase de diagnóstico del proceso se han empleado técnicas de descubrimiento de proceso. El descubrimiento de procesos consiste en determinar modelos de procesos representativos del comportamiento expresado en un registro de eventos.

El descubrimiento de proceso es una tarea compleja y con disímiles retos, los cuales influyen en la calidad de los modelos descubiertos. El ruido, las tareas duplicadas y la ausencia de información son rasgos del registro de eventos que afectan los resultados de los algoritmos de descubrimiento (De Weerdt 2012).

Las características anteriormente descritas son típicas de procesos poco estructurados. Este tipo de procesos generan registros de eventos a partir de los cuales se obtienen modelos estereotipados como espagueti (van der Aalst 2011). Una solución para el análisis de estos procesos poco estructurados es descubrir variantes de modelos alternativos del mismo proceso, denominados variantes de proceso (Pérez-Alfonso 2015).

Las variantes de procesos, se construyen automáticamente a partir de la combinación de los comportamientos del registro de eventos, expresados en las sucesiones indirectas entre las actividades del proceso. Los registros de eventos provenientes de entornos reales suelen presentar ruido y ausencia de información. Estas características se traducen en múltiples sucesiones indirectas diferentes, dificultando la identificación de los patrones de control de flujo por parte de los algoritmos de descubrimiento.

Para el descubrimiento de variantes alternativas del proceso es necesario analizar todo el comportamiento del registro de eventos. Dentro de los comportamientos que aparecen con mayor frecuencia están las sucesiones indirectas. La cantidad de sucesiones indirectas a procesar impacta en la complejidad del algoritmo de descubrimiento de variantes. Esto se convierte en una limitación para obtener patrones de control de flujo del proceso en términos de tiempo y requerimientos computacionales.

La identificación de patrones de control de flujo permite determinar las actividades que se realizan sincrónicamente, los bloques de actividades que se repiten, el orden en que se ejecutan determinadas actividades y otros comportamientos relevantes en el proceso. El reconocimiento de estos patrones de control de flujo posibilita mejorar la comprensión del funcionamiento general del proceso y su estructura.

A partir de la problemática antes descrita se define el siguiente **problema a resolver:**

El número de sucesiones indirectas entre las tareas presentes en el registro de eventos aumenta el tiempo de descubrimiento de variantes de proceso de un negocio.

A partir del problema a resolver planteado, se define como **objeto de estudio** la Minería de Procesos.

Para dar solución al problema a resolver se plantea el siguiente **objetivo general:** Desarrollar un algoritmo que disminuya el tiempo requerido para construir variantes de proceso a partir de registros de eventos independientemente del número de sucesiones indirectas entre las tareas.

Por tanto, el **campo de acción** se basa en el Diagnóstico de procesos.

Del objetivo general antes expuesto se desglosan los siguientes **objetivos específicos:**

1. Realizar el marco teórico referencial de la investigación, relacionado con los enfoques de solución utilizados por los algoritmos de descubrimiento de proceso ante registros de eventos con alto número de sucesiones indirectas entre las tareas.

2. Diseñar un algoritmo para construir variantes de proceso a partir de registros de eventos con alto número de sucesiones indirectas entre las tareas.

3. Validar el algoritmo diseñado a partir de experimentos con su implementación y registros de eventos reales.

**Posibles resultados:**

Un algoritmo que disminuya el tiempo requerido para construir variantes de proceso a partir de registros de eventos con alto número de sucesiones indirectas entre las tareas. Una implementación del algoritmo propuesto.

Para dar solución a los objetivos planteados se establecen las siguientes **tareas de la investigación**:

- Análisis de los principales conceptos y trabajos relacionados con el descubrimiento del proceso.
- Caracterización de los patrones de control de flujo y su representación en un registro de eventos.
- Análisis del comportamiento de los algoritmos existentes para el descubrimiento de procesos en registros de eventos con alto número de sucesiones indirectas entre las tareas.
- Diseño de un algoritmo para construir variantes de proceso a partir de registros de eventos con alto número de sucesiones indirectas entre las tareas.
- Implementación del algoritmo diseñado permitiendo su integración a un marco de trabajo existente para la minería de procesos.
- Selección de registros de eventos con alto número de sucesiones indirectas entre las tareas.
- Evaluación del algoritmo, a través de su implementación, con los registros de eventos seleccionados.

En la consecución de los objetivos trazados se utilizaron los siguientes métodos científicos:

- **Analítico-Sintético**: permitió realizar el estudio teórico de la investigación, facilitando el análisis de documentos y la extracción de los elementos fundamentales acerca de la utilización de la minería de procesos en entornos reales.
- **Histórico-Lógico**: permitió realizar un estudio sobre las principales herramientas empleadas en el desarrollo de la minería de procesos, las tendencias del uso actual de las mismas en el mundo y en proyectos de la universidad, con el fin de seleccionar las más apropiadas para darle cumplimiento al objetivo general de la presente investigación.
- **Hipotético-deductivo**: se utiliza para guiar la investigación desde el mantenimiento del problema hasta la verificación de la solución a partir de las validaciones, orientando la secuencia lógica de las tareas que se realizan.
- **Experimentación**: se emplea para verificar la mejora de tiempo, así como la consistencia entre los modelos descubiertos por el algoritmo y los modelos de proceso existente del proceso. Se utiliza para verificar la consistencia entre los resultados obtenidos con la

aplicación del algoritmo propuesto y los resultados de la aplicación de técnicas de diagnóstico de proceso similares sobre el mismo registro de eventos.

En el presente documento puede encontrarse un resumen, una introducción, tres capítulos y conclusiones. A continuación, se resume el contenido de los capítulos:

**Capítulo 1**, *"Fundamentación teórica"*, se introducirán los principales conceptos asociados al objeto de estudio: la minería de procesos. Se brinda una descripción del entorno en el cual se desarrolla la problemática y se analizan las posibles soluciones. Se realiza un estudio de la Minería de Variantes como método para la obtención de variantes de proceso, así como sus fortalezas y debilidades. Se realiza un estudio de los algoritmos de descubrimiento de procesos existentes y se analiza cómo estos abordan las limitaciones de la Minería de Variantes. Se presentan las principales características del marco de trabajo para la minería de procesos ProM.

**Capítulo 2**, *"Algoritmo para disminuir la complejidad temporal de la Minería de Variantes"*, se propone un algoritmo para obtener variantes de proceso. Se describe la implementación del algoritmo anteriormente mencionado.

**Capítulo 3**, *"Validación del algoritmo"*, se analiza la calidad de los modelos en el ámbito del chequeo de conformidad, una de las 3 tareas de la minería de procesos. Se describen las dimensiones de calidad para la evaluación de los modelos y se escogen las métricas a evaluar para cada una. Se estudia la herramienta utilizada para la validación denominada CoBeFra. Se describen las pruebas de tiempo y calidad realizadas al algoritmo. Se seleccionan para la evaluación registros de eventos con diferentes números de sucesiones indirectas. Se evalúan los resultados de los modelos obtenidos a partir del algoritmo propuesto para los registros de eventos seleccionados.

# CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se describen los conceptos asociados a la minería de procesos con la gestión de procesos de negocio. Se describe el entorno en el cual se desarrolla la problemática y se analizan las posibles soluciones. Se realiza un estudio crítico de las técnicas de minería de procesos, con énfasis en el Descubrimiento de Procesos. Se analizan diferentes algoritmos, sus ventajas y limitaciones, así como su comportamiento ante registros de eventos con las características descritas en la introducción. Se presentan las principales características del marco de trabajo para la minería de procesos ProM.

## 1.1. Minería de Procesos

La minería de procesos es una disciplina científica cuyo fin es descubrir, monitorear y mejorar los procesos reales. Se basa en la extracción de conocimiento sobre los procesos a partir de los registros de eventos. Dentro de la minería de procesos se realizan tres grandes tareas: el descubrimiento de procesos, la verificación de conformidad y el mejoramiento de modelos.

Un registro de eventos es el resultado del almacenamiento por parte de los sistemas de información de las ejecuciones de un proceso, o sea, un registro de las actividades de un proceso ejecutadas en un período determinado de tiempo. Las ejecuciones del proceso se denominan trazas. La minería de procesos tiene definidos dos estándares para formalizar la estructura de los registros de eventos: MXML y XES. MXML surgió en el 2003 y fue el primer estándar adoptado por la herramienta ProM. El Flujo Extensible de Eventos (*eXtensibleEventStream*, XES por sus siglas en inglés) reemplaza a MXML en el 2010 como el nuevo formato para la minería de procesos independiente de la herramienta. El formato XES es un estándar XML para los registros de eventos y fue tomado por la IEEE Task Force on Process Mining, como el formato por defecto para el intercambio de registros de eventos (van der Aalst 2011). Los datos requeridos por el formato XES son: la marca de tiempo, identificador, nombre de la actividad y recurso.

Para descubrir modelo de procesos con calidad se asume que el registro de eventos contiene una muestra distintiva del comportamiento, pero existen dos fenómenos que pueden hacerlo menos representativo: ruido y ausencia de información, definiciones 3 y 4 tomadas de (van der Aalst 2011) y (Yzquierdo-Herrera 2012) respectivamente.

**Definición 3.** *El r*uido es el c*omportamiento contenido en el registro de eventos que es raro, poco frecuente y no es representativo del comportamiento típico del proceso.*

El impacto del ruido en las técnicas de minería de procesos provoca desde la identificación de patrones incorrectos en el registro de eventos hasta la obtención de modelos complejos poco

estructurados (Ciccio, Mecella y Mendling 2015; Mitsyuk y Shugurov 2014). Para resolver estos problemas el ruido suele ser descartado, ya sea durante el pre-procesamiento del registro de eventos o durante el descubrimiento del modelo de proceso.

Para la extracción de ruido en el pre-procesamiento se identifican las trazas anómalas. Para esta identificación se utiliza un umbral con base en la frecuencia de las trazas en el registro de eventos (Bezerra y Wainer 2012). Los resultados obtenidos dependen de la distribución de frecuencia de las trazas, por lo que pueden ser descartadas trazas sin ruido. Además, en una misma traza puede existir ruido y comportamiento que no lo es.

Tratar el ruido a través del pre-procesamiento tiene como principal limitación que, aunque se puede identificar los comportamientos o las trazas poco frecuentes no se puede determinar si estos representan comportamiento típico del proceso. Una aproximación a este comportamiento es el expresado en el modelo descubierto. Por tanto, es preferible tratar el ruido durante el descubrimiento del modelo del proceso.

**Definición 4.** *La ausencia de información es la falta en las trazas de eventos relativos a la ejecución de actividades, que dificulta la identificación de patrones de control de flujo del proceso. Si las trazas no contienen ningún evento de alguna actividad a este tipo de actividad se le denominará actividad invisible.*

Otro reto para las técnicas de minería de procesos es la ausencia de eventos en las trazas. Es posible que, en una o varias de las actividades que conforman el proceso no aparezca ningún evento, porque no han sido informatizados o el sistema no registra su ocurrencia. Este fenómeno se denomina ausencia de información (Yzquierdo-Herrera, Silverio-Castro y Lazo-Cortés 2013). Cuando la cantidad de eventos presente es insuficiente para descubrir algunos de los patrones de control de flujo del proceso se considera que el registro de eventos está incompleto (van der Aalst 2011).

La completitud o que el registro de eventos contenga todos los posibles caminos de ejecución del proceso es altamente improbable. Los modelos de procesos suelen permitir que se genere un alto número de comportamientos. Un paralelismo de $n$ actividades, o lo que es lo mismo, las $n$ actividades deben ejecutarse, pero en cualquier orden genera $n!$ secuencias de ejecuciones posibles; en el caso de un ciclo este número pudiera ser infinito (van der Aalst 2011). Las técnicas de minería de procesos deben asumir que no todo el comportamiento está en el registro de eventos, debido a la baja probabilidad de que exista completitud total.

## 1.2. Descubrimiento de procesos

Dentro de la minería de procesos el descubrimiento de procesos es la tarea que se ocupa de construir automáticamente un modelo de proceso que represente el comportamiento de las ejecuciones del proceso a partir de un registro de eventos. El descubrimiento posee algoritmos con diferentes enfoques siendo así el área más atendida dentro de la minería de procesos. Entre estos enfoques se encuentra el genético y el difuso (van der Aalst 2011).

Un algoritmo de descubrimiento es una función que mapea un registro de eventos hacia un modelo de proceso. Estos modelos descubiertos pueden ser representados mediante múltiples notaciones, entre ellas se encuentran las Redes de *Petri* y Redes de Flujo de Trabajo. Las redes de *Petri* son el lenguaje de modelado de procesos más viejo y mejor investigado que permite concurrencias. La notación gráfica las redes de *Petri* es intuitiva y simple, las redes de *Petri* son ejecutables y muchas técnicas de minería de procesos pueden ser utilizadas para analizarlas (van der Aalst 2011). Dentro de los algoritmos de descubrimiento existe la Minería de Variantes que devuelve varias variantes alternativas de modelos de procesos.

## 1.3. Minería de Variantes

La Minería de Variantes es un método que tiene como objetivo obtener información relevante para el diagnóstico de procesos de negocio a partir de registros de eventos con ruido y ausencia de información. Para lograr este objetivo se construyen diferentes variantes de modelos de proceso (Definición 7), a partir de la descomposición en subprocesos (Pérez-Alfonso 2015). En este ámbito se considera que un subproceso es:

**Definición 5.** *Un subproceso es una encapsulación de las actividades del negocio que representa una unidad de trabajo lógica y coherente. Los subprocesos tienen sus propios atributos y metas, pero contribuyen a alcanzar la meta general del proceso. Un subproceso es también un proceso y la mínima expresión de un subproceso es una actividad* (Yzquierdo-Herrera 2012).

La Minería de Variantes posee un enfoque diferente al de otras técnicas de minería de procesos, al proponer varias descomposiciones alternativas para el mismo subproceso, utilizando diferentes operadores de control de flujo. Esto permite controlar el impacto estructural del ruido y la ausencia de información en la construcción de las alternativas. Las alternativas se construyen al descartar o considerar comportamientos poco frecuentes que están contenidos en el registro de eventos. También, en dicha construcción se asumen comportamientos ausentes del registro de eventos debido a situaciones de ausencia de información. Las diferentes alternativas de descomposición que pueden existir en cada subproceso son representadas como variantes de proceso en un Árbol de Variantes

(Pérez-Alfonso 2015). En la Minería de Variantes solo se consideran los patrones descritos en la Definición 6:

**Definición 6.** Patrones de control de flujo:

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

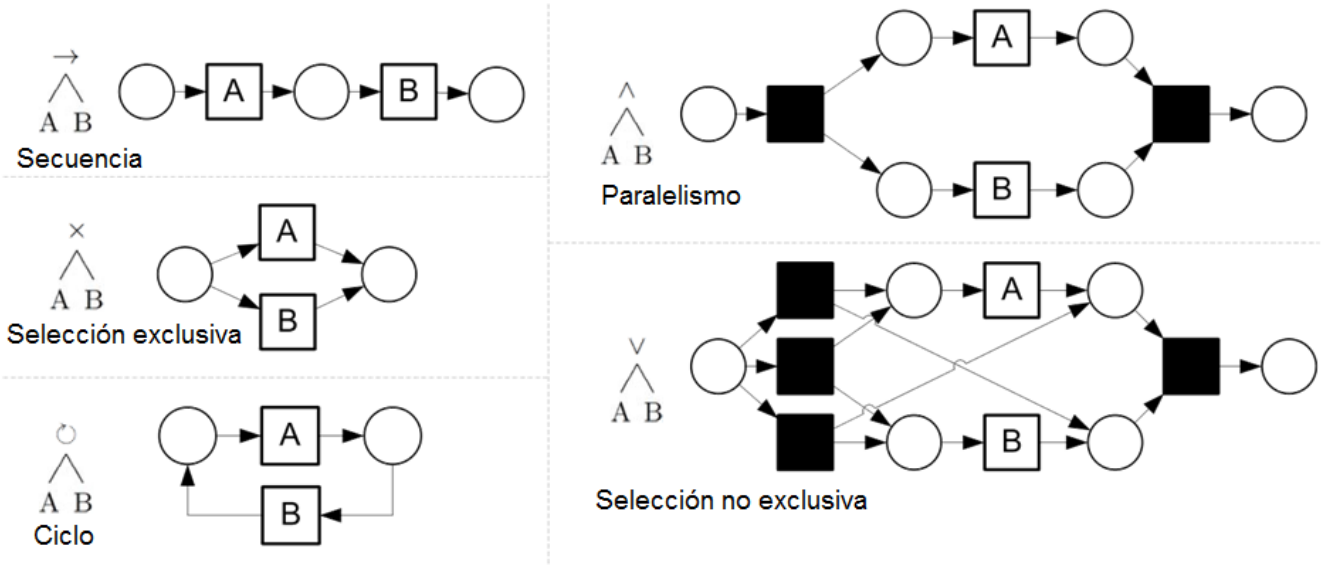


***Figura 3:*** *Representación en Red de Petri de los patrones de control de flujo* (Buijs, van Dongen y van der Aalst, W.M.P. 2012).

**Definición 7.** *Las variantes de un modelo de proceso o variantes de proceso, son modelos de un proceso que describen el mismo proceso de negocio y poseen diferencias estructurales. Las diferencias están dadas por los patrones de control de flujo que se utilizan en secciones equivalentes del proceso y la presencia de determinadas actividades* (Pérez-Alfonso 2015).

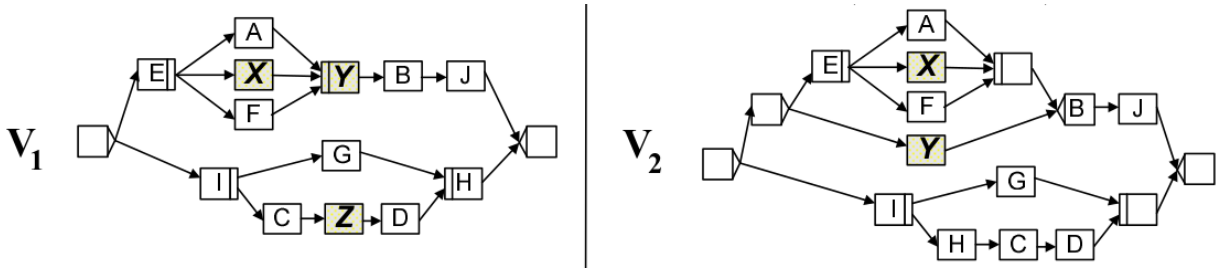


***Figura 4:*** *Variantes de modelo de procesos* (Li, Reichert y Wombacher 2011).

En la Figura 4 se pueden apreciar las variantes V1 y V2 de un proceso de negocio. En la V1, la actividad Y se muestra en secuencia con el subproceso conformado por las actividades E, A, X y F; por otro lado en V2, la actividad Y aparece en selección exclusiva con el subproceso antes mencionado. Además la actividad Z, aparece en V1, pero no en V2.

Las variantes de proceso se representan en el árbol de variantes, el cual está compuesto por dos tipos de nodos: de subproceso y patrón. Un nodo subproceso representa un subproceso y posee tantos nodos patrones de hijos, como posibles descomposiciones se hayan identificado para el subproceso. 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 se refiere a todo el proceso. Los nodos hojas son siempre nodos de tipo subproceso (Pérez-Alfonso 2015).

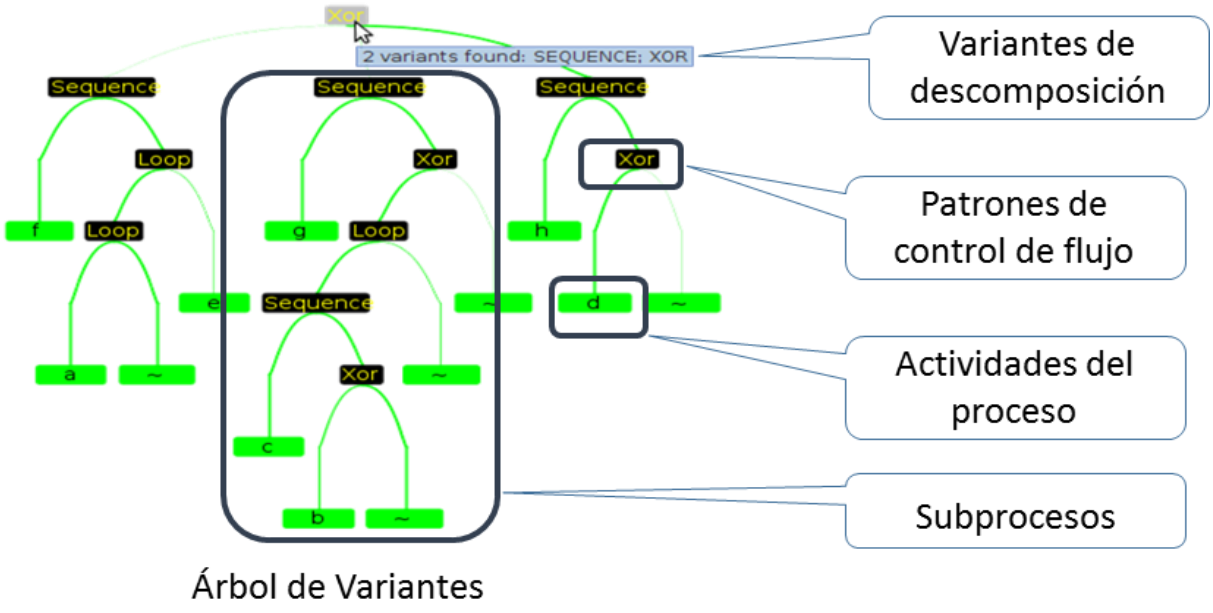


***Figura 5:*** *Ejemplo de Árbol de variantes donde se pueden observar sus componentes* (Pérez-Alfonso 2015).

La técnica de Minería de Variantes está conformada por tres etapas que se ejecutan secuencialmente: pre-procesamiento del registro de eventos, extracción de comportamiento y obtención de variantes e información de diagnóstico (Pérez-Alfonso 2015) como se muestra en la Figura 6.

El pre-procesamiento tiene como objetivo preparar el registro de eventos para el diagnóstico de variantes. Se le asigna una letra a cada evento contenido en el registro, se unifican las trazas cuyas secuencias coinciden y las secuencias resultantes de la agrupación son ordenadas descendentemente, por su frecuencia de aparición, para expresar su importancia relativa dentro del proceso (Pérez-Alfonso 2015).

El propósito de la segunda etapa es la extracción de comportamientos representativos de los patrones de control de flujo, a partir del registro de eventos pre-procesado. Para la técnica son de relevante importancia los comportamientos definidos por (Pérez-Alfonso et al. 2015):

**Definición 8.** *Comportamiento del proceso: denotemos como* $l_i \subseteq L_P$ *a la sección del registro de eventos relacionada al subproceso* $s_i \in S_P$*. El comportamiento del patrón de control de flujo* $w_j$ *para el subproceso* $s_i$ : $B_{w_j s_i} | \, w_j \in W$*, está compuesto por las posibles:*

- *Sucesiones directas (SD): se denomina sucesión directa entre la actividad **a** y la actividad **b** a la existencia en alguna traza de la secuencia **ab**.*
- *Sucesiones indirectas (SI): se denomina sucesión indirecta entre la actividad **a** y la actividad **b**, a la aparición en alguna traza de la actividad **a** seguida, inmediatamente o no, por la actividad **b**.*
- *Eventos que inician y/o finalizan trazas: son los eventos que aparecen de primeros y/o últimos en alguna de las trazas.*
- *Eventos repetidos en las trazas: son los eventos que aparecen en más de una ocasión en una misma traza.*

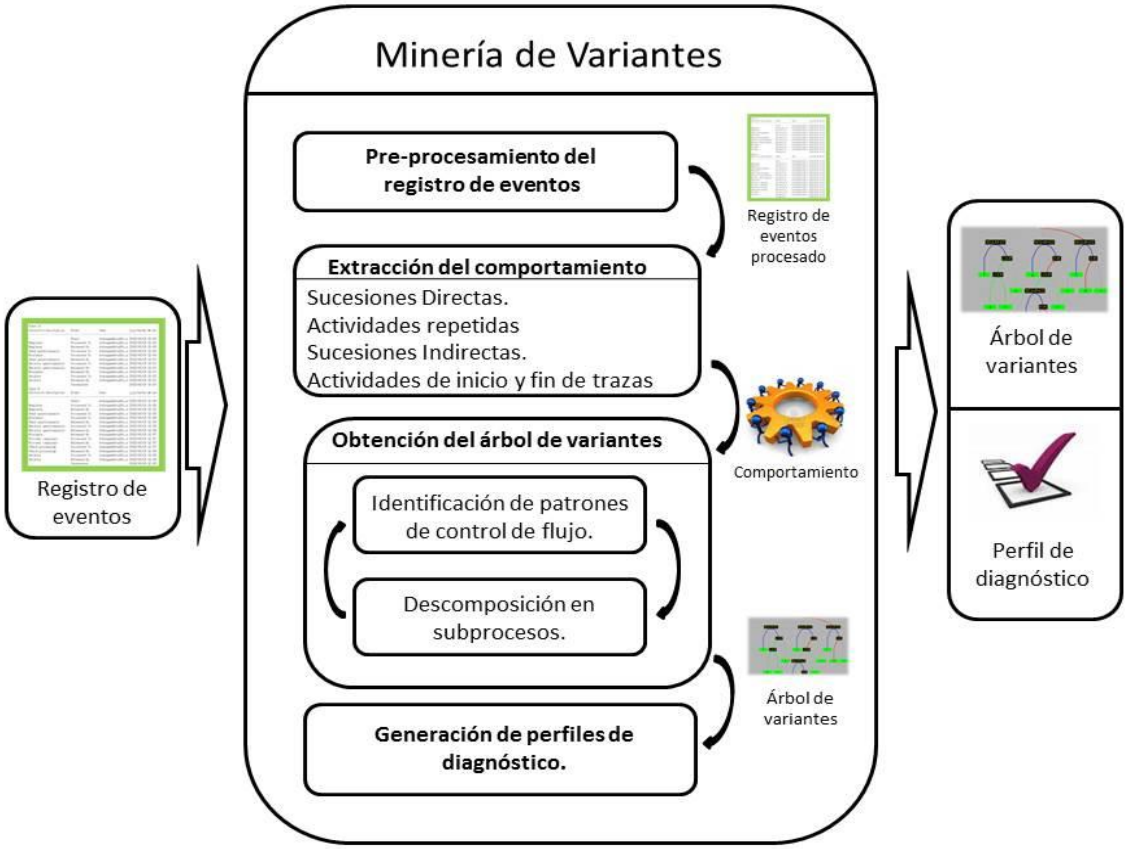


***Figura 6:*** *Minería de Variantes, entrada, salidas y sus etapas* (Pupo-Hernández y López-Jiménez 2014).

Por cada comportamiento extraído se obtiene la frecuencia de su aparición en las trazas codificadas y se ordenan en función de estas frecuencias. Esta frecuencia es utilizada para establecer cuán representativo es un comportamiento del proceso que lo genera. Los comportamientos extraídos en esta fase se combinan durante la obtención de variantes de descomposición. La extracción de comportamiento permite identificar, con mayor precisión que a nivel de traza, los elementos de ejecución del proceso. La baja frecuencia de ejecución en el comportamiento sugiere que deban ser considerados como ruido. Los comportamientos extraídos son la base para identificar los operadores de control de flujo que deben ser utilizados para la descomposición de cada subproceso (Pérez-Alfonso 2015).

En la tercera etapa se obtienen las variantes de descomposición del proceso para su representación en un Árbol de Variantes. Las variantes se obtienen a partir del procesamiento de los comportamientos extraídos en la fase anterior y la asunción de ciertos comportamientos ausentes del registro de eventos. En esta etapa se realizan dos tareas iterativamente, la descomposición en subprocesos y la identificación de patrones de control de flujo. La obtención de variantes

considerando patrones de control de flujo constituye el núcleo del método Minería de Variantes y uno de sus aportes prácticos (Pérez-Alfonso 2015).

Para la obtención de variantes se sigue un enfoque "divide y vencerás". Se parte del proceso como un todo y se identifican los operadores de control de flujo que pueden ser utilizados para su descomposición o división en varios subprocesos. La descomposición que se logra con cada operador se convierte en una variante para ese subproceso. A los subprocesos obtenidos con cada descomposición se les realiza el mismo procedimiento. Por cada variante obtenida se extrae la información de diagnóstico asociada a la misma, la cual es guardada en el Perfil de Diagnóstico (Definición 9). En el Perfil de Diagnóstico se agrupa la información de diagnóstico sobre cada nodo de tipo operador o patrón de control de flujo del Árbol de Variantes descubierto (Pérez-Alfonso 2015).

**Definición 9.** *El Perfil de Diagnóstico* $(D)$ *de un Árbol de Variantes* $VT$ *agrupa la información de diagnóstico asociada a la obtención de las variantes que conforman a* $VT$*. Sea la tupla* $d_i^w = [l_i, n_i^w, \theta_i^w, f, \rho]$ *la información de diagnóstico de cada nodo de tipo operador* $pn_i^w$ *en* $VT$*. Donde* $n_i^w$ *es el comportamiento descartado y* $\theta_i^w$ *es el comportamiento asumido en la obtención de la variante que representa* $pn_i^w$*. El valor de f es una estimación de aptitud que expresa la relación entre* $n_i^w$ *y* $\beta_i$ *(el comportamiento expresado en* $l_i$*). ρ es una estimación de precisión que expresa la relación entre* $\theta_i^w$ *y* $\vartheta_i^w$ *(el comportamiento del proceso en* $s_i$ *para el patrón que representa w y que no forma parte de* $\beta_i$*).* $l_i$ *es la sección del registro de eventos relativa al subproceso cuya descomposición representan* $pn_i^w$ (Pérez-Alfonso 2015).

Los comportamientos descartados y asumidos en las descomposiciones alternativas identificadas por cada subproceso se incluyen en el Perfil de Diagnóstico. Estos comportamientos pueden ser anomalías o desviaciones durante la ejecución del proceso y reflejar situaciones de ausencia de información. La presentación de ambos tipos de comportamiento responde directamente a los intereses del diagnóstico y pretende contribuir a la comprensión del proceso en la dimensión pragmática (Pérez-Alfonso 2015).

### 1.3.1. Análisis de la complejidad algorítmica de la Minería de Variantes

La Minería de Variantes tiene una complejidad algorítmica de $O(P^a * B(a))$. Donde P es la cantidad de patrones oscilando de 0 a 5, $a$ es el número de actividades y $B(a)$ es el número de Bell o cantidad de particiones que se pueden hacer de un conjunto de $a$ elementos (Rosen y Krithivasan 1999). En este caso la cantidad de particiones son las diferentes descomposiciones en subprocesos que se pueden hacer por cada patrón sobre un conjunto de $a$ actividades que tiene el subproceso que se está analizando.

En la Tabla 1 se muestran los valores que van tomando los factores de la fórmula de la complejidad para diferentes cantidades de patrones y actividades. Se puede observar que los valores del factor $P^a$ para 5 patrones inicialmente son mayores que los de $B(a)$. Pero este último tiene un crecimiento mayor y ya para 22 actividades sobrepasa el otro factor, siendo así el que más influye en la complejidad algorítmica cuando se identifican 5 patrones para registros de eventos con más de 22 actividades.

***Tabla 1:*** *Complejidad algorítmica de la Minería de variantes para 5 patrones de control de flujo.*

| a | $B(a)$ | $2^a$ | $3^a$ | $4^a$ | $5^a$ |
|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 2 | 3 | 4 | 5 |
| 2 | 2 | 4 | 9 | 16 | 25 |
| 3 | 5 | 8 | 27 | 64 | 125 |
| 4 | 15 | 16 | 81 | 256 | 625 |
| 5 | **52** | **32** | 243 | 1024 | 3125 |
| 6 | 203 | 64 | 729 | 4096 | 15625 |
| 7 | 877 | 128 | 2187 | 16384 | 78125 |
| 8 | 4140 | 256 | 6561 | 65536 | 390625 |
| 9 | **21147** | 512 | **19683** | 262144 | 1953125 |
| 10 | 115975 | 1024 | 59049 | 1048576 | 9765625 |
| 11 | 678570 | 2048 | 177147 | 4194304 | 48828125 |
| 12 | 4213597 | 4096 | 531441 | 16777216 | 2.44E+08 |
| 13 | 27644437 | 8192 | 1594323 | 67108864 | 1.22E+09 |
| 14 | 1.91E+08 | 16384 | 4782969 | 2.68E+08 | 6.1E+09 |
| 15 | **1.38E+09** | 32768 | 14348907 | **1.07E+09** | 3.05E+10 |
| 16 | 1.05E+10 | 65536 | 43046721 | 4.29E+09 | 1.53E+11 |
| 17 | 8.29E+10 | 131072 | 1.29E+08 | 1.72E+10 | 7.63E+11 |
| 18 | 6.82E+11 | 262144 | 3.87E+08 | 6.87E+10 | 3.81E+12 |
| 19 | 5.83E+12 | 524288 | 1.16E+09 | 2.75E+11 | 1.91E+13 |
| 20 | 5.17E+13 | 1048576 | 3.49E+09 | 1.1E+12 | 9.54E+13 |
| 21 | 4.75E+14 | 2097152 | 1.05E+10 | 4.4E+12 | 4.77E+14 |
| 22 | **4.51E+15** | 4194304 | 3.14E+10 | 1.76E+13 | **2.38E+15** |
| 23 | 4.42E+16 | 8388608 | 9.41E+10 | 7.04E+13 | 1.19E+16 |
| 24 | 4.46E+17 | 16777216 | 2.82E+11 | 2.81E+14 | 5.96E+16 |
| 25 | 4.64E+18 | 33554432 | 8.47E+11 | 1.13E+15 | 2.98E+17 |
| 26 | 4.96E+19 | 67108864 | 2.54E+12 | 4.5E+15 | 1.49E+18 |

Pero en la realidad es más frecuente la descomposición en 3 o menos patrones. En este caso el Número de Bell sobrepasa al otro factor ante registros de eventos con 9 actividades o más como se observa subrayado en la Tabla 1. Esto trae como consecuencia que como en la mayoría de los casos

se descomponen los subprocesos en 3 o menos patrones, el $B(a)$ tenga mayor impacto en la complejidad algorítmica.

El número de Bell representa la cantidad máxima de descomposiciones que se pueden realizar de un subproceso. Nunca es explorado el máximo de descomposiciones, ya que la cantidad real de estas depende del comportamiento que se procesa. En un registro de eventos, el máximo número de comportamiento que se puede encontrar siendo $a$ la cantidad de actividades es: $a * (a - 1)$ para las sucesiones directas (cuando todas están en paralelo), $a * a$ para las sucesiones indirectas y $a$ para las actividades de inicio y fin de traza. Por tanto, las sucesiones indirectas son la mayor parte del comportamiento que se procesa, siendo así las que más influyen en el factor de complejidad $B(a)$.

Teniendo lo anterior en consideración y como no se puede disminuir la cantidad de actividades en un registro de eventos, la forma más conveniente de reducir el tiempo de ejecución de la Minería de Variantes es disminuyendo la cantidad de comportamiento procesado. De este comportamiento las sucesiones indirectas son las que más inciden sobre el número de Bell. Para encontrar una solución a este problema se hace un análisis de cómo los principales algoritmos de descubrimiento de procesos tratan el comportamiento presente en los registros de eventos.

## 1.4. Algoritmos de descubrimiento

Existen varios algoritmos de descubrimiento de procesos. Ante registros de eventos estos algoritmos extraen el comportamiento. La cantidad de comportamiento extraído aumenta en dependencia del ruido y la ausencia de información presente en el registro de eventos. A continuación, se hace un análisis de algunos de los algoritmos de descubrimiento de procesos y cómo hacen frente a estos problemas.

El algoritmo *Alpha* se enfoca en el control de flujo, analizando solamente el orden de las actividades, ignora recursos, marcas de tiempo, ID del caso y otros datos. A través del registro de eventos se realiza una matriz de huellas para determinar los patrones secuencia, selección exclusiva, paralelismo y con ellos construir una red de *Petri* (en la Figura 7 se puede ver un ejemplo de una red de *Petri* sencilla). Fue uno de los primeros algoritmos capaces de descubrir concurrencia presente en un registro de eventos. Sin embargo, presenta varias limitaciones como con los estados de transición implícitos. No maneja correctamente los ciclos de longitud 1 y 2, pudiendo dejar transiciones sin relacionarse con el resto de la red de *Petri*. Nunca generará una red con dos transiciones que tengan el mismo nombre. Puede producir modelos con puntos muertos (*deadlocks*). No es capaz de manejar el ruido y la ausencia de información y presenta problemas con las dependencias no locales o sucesiones indirectas (van der Aalst, W.M.P., Weijters y Maruster 2004).

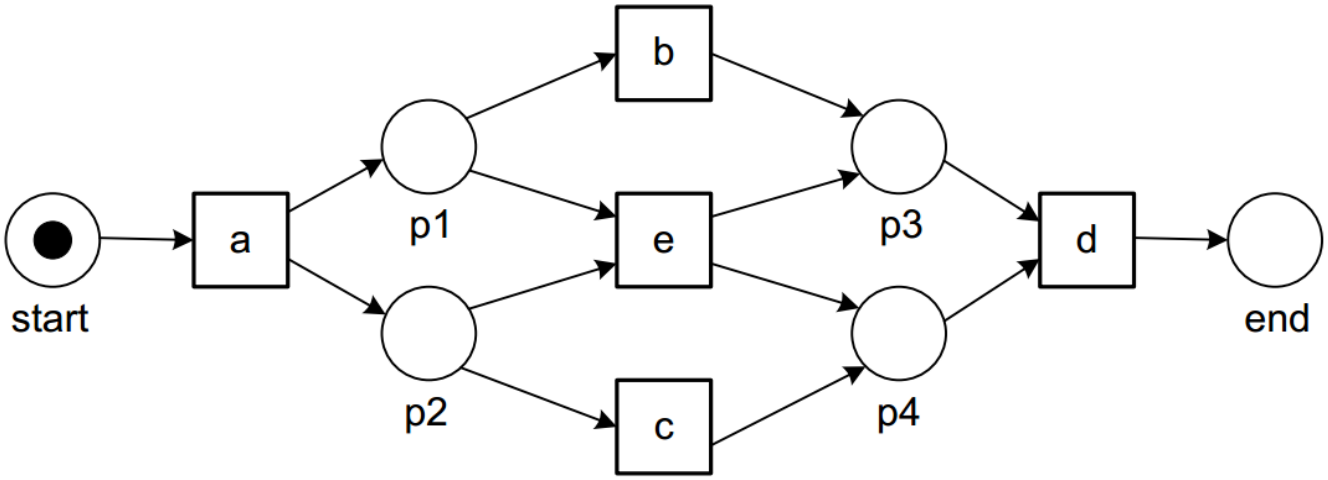


***Figura 7:*** *Representación de un proceso en una red de Petri, obtenida a partir del algoritmo Alpha* (van der Aalst 2011).

Muchos otros algoritmos cubren algunas de las limitaciones presentes en el *Alpha*. La minería difusa o *Fuzzy Miner* es un algoritmo que construye un gráfico del proceso como se muestra en la Figura 8. Este algoritmo no utiliza patrones de control de flujo. Como resultado se obtiene un modelo simplificado basado en gráficos. Este modelo es capaz de proporcionar una vista de alto nivel de un proceso abstrayéndose de los detalles no deseados (Günther y van der Aalst, W.M.P. 2007).

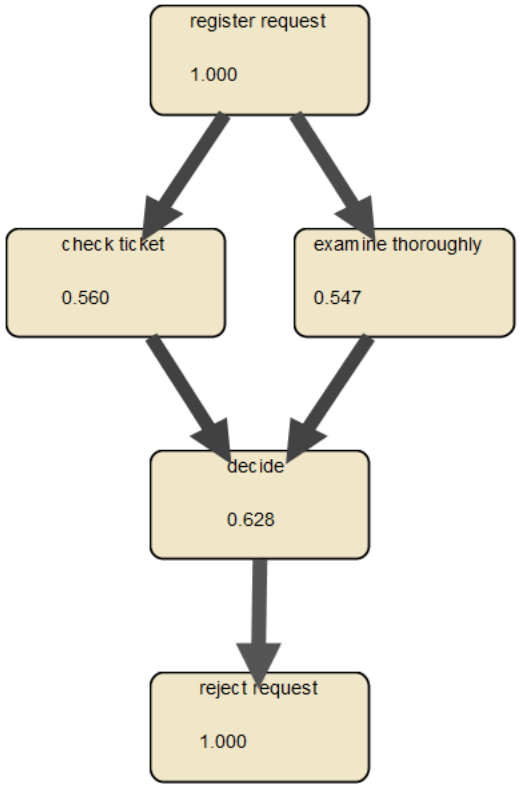


***Figura 8:*** *Representación de un gráfico del proceso, obtenida a partir del algoritmo Fuzzy Miner.*

El *Fuzzy Miner* identifica dos métricas fundamentales: significancia y correlación para simplificar y visualizar apropiadamente procesos complejos y menos estructurados. La primera puede ser determinada para actividades y las relaciones binarias de precedencia sobre ellas. Mide la importancia relativa del comportamiento especificando el nivel de interés que tenemos en eventos o su ocurrencia uno después de otro. Una forma para medir la significancia es la por la frecuencia por ejemplo, eventos o relaciones de precedencia que son observados más frecuentemente son supuestamente más significativos. Por otra parte, la correlación es relevante solamente para relaciones de precedencia sobre eventos ya que mide qué tan relacionados están dos eventos que se siguen uno a otro. Para medir la correlación se puede determinar la coincidencia de atributos asociados a dos eventos que se siguen uno al otro o comparar la similitud de sus nombres de eventos ya que se asume que mientras más correlacionados dos eventos estén más cantidad de información comparten (Günther y van der Aalst, W.M.P. 2007).

Basados en estas dos métricas la minería difusa utiliza un enfoque para la simplificación de procesos en el modelado. Cuando se crea el modelo simplificado se preserva el comportamiento altamente significativo, se remueve el menos significativo y poco correlacionado y se añade en grupos el menos significativo, pero altamente correlacionado. Este enfoque puede reducir o enfocar el comportamiento mostrado, permitiendo emplear el concepto de énfasis destacando el comportamiento más significativo en el modelo simplificado (Günther y van der Aalst, W.M.P. 2007).

Uno de los algoritmos más utilizados para el descubrimiento de procesos es el *Heuristic Miner*. Este utiliza como entrada un registro de eventos, del cual solo utiliza el nombre de la actividad, identificador del caso y la marca de tiempo. La marca de tiempo es empleada para calcular el orden de los eventos dentro del caso, el orden de ellos entre casos no es importante. Considera las frecuencias de los eventos y las secuencias, los caminos poco frecuentes no son incorporados al modelo. Adicionalmente tiene una opción configurable por el usuario para tratar sucesiones indirectas mediante un umbral (Weijters, van der Aalst, W.M.P. y de Medeiros 2006).

El objetivo del *Heuristic Miner* es extraer una red casual (*C-net*) del registro de eventos. Para ello primero construye un grafo de dependencia donde cada nodo de este corresponde a un a un set de actividades y los arcos del grafo de dependencia representan las relaciones entre las actividades. En la *C-net* hay una única actividad de inicio $a_0$ y otra de fin $a_n$ que pueden ser creadas insertando un evento artificial de inicio y uno de fin a cada traza. Se asume que todas las actividades en el grafo de dependencia están en algún camino entre $a_0$ y $a_n$, se eliminan aquellas que no lo estén. Por tanto, una vez construido el grafo de dependencia se tiene la estructura principal de la *C-net* la cual muestra la perspectiva de control de flujo permitiendo a los usuarios enfocarse en los principales procesos (Figura 9) (van der Aalst 2011).

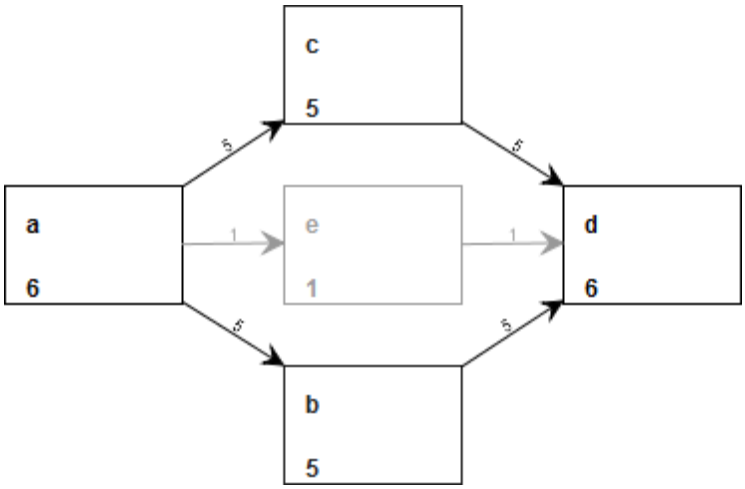


***Figura 9:*** *Representación de un proceso a través de una C-net, obtenida a partir del algoritmo Heuristic Miner.*

La técnica del *Inductive Miner* (IM) permite descubrir modelos en forma de árboles de proceso (Figura 10). Un árbol de proceso es una representación de una red de flujo de trabajo. Las hojas son actividades individuales y los nodos que no son hojas son operadores que describen cómo se combinan sus hijos. Se consideran los siguientes operadores: →, ×, ∧, ↻. El operador → denota la secuencia entre sus hijos, × representa la relación de exclusión mutua, ∧ se utiliza para representar el paralelismo y ↻ es el lazo (Leemans, Sander J. J., Fahland y van der Aalst, W.M.P. 2013).

IM trabaja de manera recursiva, seleccionando el operador raíz que mejor se ajusta a un registro de eventos, dividiendo el registro en dos conjuntos disjuntos de actividades y utilizando estos conjuntos como sub-registros. Estos sub-registros son analizados de manera recursiva hasta que alguno contenga una sola actividad. IM utiliza los sistemas de transición y las regiones para dividir el registro. Para un registro de eventos se crea un grafo de sucesiones directas, en el cual cada nodo representa una actividad y las aristas entre las actividades $a$ y $b$ estarán presentes solo cuando entre $a$ y $b$ exista una relación de sucesión directa. La frecuencia de aparición de esta relación está representada como el peso de la arista (Leemans, Sander JJ, Fahland y van der Aalst, W.M.P. 2013).

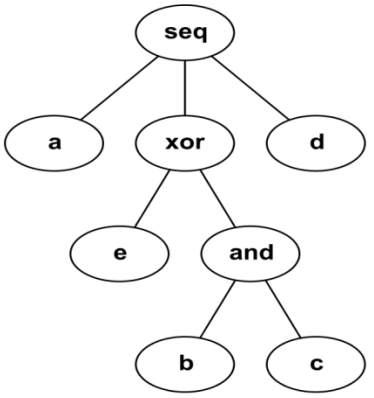


***Figura 10:*** *Representación de un árbol de proceso, obtenida a partir del algoritmo Inductive Miner.*

El algoritmo *Constructs Competition Miner* (CCM) es un algoritmo para descubrir un modelo de procesos asumiendo como enfoque que este modelo está compuesto por un conjunto de bloques como son: secuencia, selección, ciclo y paralelismo. El modelo descubierto se logra asumiendo que el proceso es una estructura anidada. Permitiendo los bloques competir entre ellos en cada nivel por la solución más viable desde lo más simple a lo más complejo. El aspecto competitivo del CCM lo hace especialmente adecuado para registros de eventos con comportamientos excepcionales y conflictivos entre sí (Redlich et al. 2014).

La motivación principal de desarrollar un algoritmo que hace que diferentes bloques compitan por la mejor solución está derivada a partir del desafío que representa el ruido o incluso los comportamientos frecuentes, pero incompatibles en los registros de eventos. En casos de incertidumbre el algoritmo debería buscar la mejor solución que soporte la mayoría del comportamiento observado. Ejemplo: secuencia, elección no libre, paralelismo o ciclos, o la combinación de estos. Otro aspecto importante del CCM es que comprende la estructura del proceso desde las relaciones globales entre cualesquiera dos actividades, por ejemplo, tiene en cuenta las relaciones entre pares de actividades que se preceden en alguna traza. Este enfoque tiene la ventaja de evitar una explosión de estados en registros de eventos correspondientes a procesos de negocio cuyas actividades estén fuertemente conectadas entre sí. Esto representa una ventaja posterior para el algoritmo de competición (Redlich et al. 2014).

El *Alpha* fue el primer algoritmo en resolver el problema de la concurrencia. Este algoritmo tenía varias limitaciones, entre ellas el no considerar ruido y ausencia de información. Estas limitaciones impulsaron el desarrollo de otros algoritmos dirigidos a estos problemas. Entre las soluciones más frecuentes al comportamiento generado por el ruido y la ausencia de información está el tratamiento por frecuencia de las trazas. Esta solución se considera en el presente trabajo para la mejora del tiempo para obtener variantes de procesos (Redlich et al. 2014).

***Tabla 2:*** *Comparación entre los algoritmos de descubrimientos analizados.*

| Algoritmo | Tratamiento de sucesiones indirectas | Configurable por el usuario | Herramienta |
|---|---|---|---|
| *Alpha* | No | No | ProM |
| *Fuzzy Miner* | Sí | Sí | ProM |
| *Heuristic Miner* | Sí | Sí | ProM |
| *Inductive Miner* | Sí | Sí | ProM |
| CCM | Sí | Sí | ProM |
| *Variants Miner* | Sí | Sí | ProM |

## 1.5. Herramientas y tecnología

Como se puede apreciar en la Tabla 2 todos los algoritmos analizados tienen implementación en ProM. Este es un marco de trabajo para el desarrollo de herramientas de minería de procesos en un ambiente estandarizado (Yang et al. 2012). Está desarrollado en Java y se encuentra disponible bajo licencia GPL1.0. ProM está concebido para admitir la adición de complementos y de esta manera posibilitar el desarrollo de nuevos algoritmos y técnicas en el campo de la minería de procesos. Los complementos necesitan determinada cantidad de parámetros de entrada y producen uno o varios objetos de salida. Los parámetros de entrada pueden ser registros de eventos u objetos obtenidos a partir del procesamiento realizado por otros complementos. Mientras que los objetos de salida obtenidos pueden ser empleados como parámetros de entrada de otros complementos. Este marco de trabajo cuenta con más de 600 complementos, cada uno de los cuales posibilita realizar diferentes análisis (van der Aalst, W.M.P. 2013). Las herramientas desarrolladas en ProM han sido empleadas en el análisis de procesos provenientes de diferentes dominios entre los que se encuentran gubernamental, hospitalario y sistemas ERP (van der Aalst, W.M.P. 2011).

Para la modificación del *plugin Variants Miner* se empleó el NetBeans como entorno de desarrollo. Este entorno facilita el desarrollo de aplicaciones en lenguaje Java, PHP, C/C++, HTML, CSS JavaScript, entre otros. Es el entorno de desarrollo oficial para la versión 8 del lenguaje de programación Java. Contiene una amplia variedad de herramientas para el desarrollo de aplicaciones, entre las que se mencionan: editores de código, compiladores, depuradores (debuggers) y

analizadores de código. Posee un editor con resaltado de sintaxis y semántica, indentación de líneas y refactorización de código. NetBeans cuenta con integración a herramientas para el versionado del proyecto como Subversion, Mercurial y Git. Es una herramienta ampliamente utilizada, de código abierto y con una comunidad de desarrolladores en todo el mundo.

## 1.6. Conclusiones parciales

En este capítulo se realizó un análisis de los principales elementos de la Minería de Procesos. Debido la presencia de ruido y ausencia de información en los registros de eventos, los algoritmos de descubrimiento deben procesar una mayor cantidad de comportamiento. Dentro de los tipos comportamientos a procesar se encuentran las sucesiones indirectas, las cuáles impactan en la complejidad algorítmica de la técnica para obtener variantes de procesos.

No todos los algoritmos de descubrimiento tratan de igual forma el ruido y la ausencia de información, ni todos procesan el mismo comportamiento ni de la misma manera. Uno de los pocos elementos en común que tienen la mayoría de estos algoritmos es el agrupar las trazas provenientes del registro de eventos por sus frecuencias. Esto les permite descartar comportamientos cuya frecuencia no sea representativa con respecto al total de trazas del registro de eventos. El umbral para descartar los comportamientos usualmente es definido por el usuario.

Se seleccionó la Minería de Variantes como algoritmo a modificar siendo el único capaz de poder generar diferentes variantes de procesos de registro de eventos. De los demás algoritmos analizados se consideraron elementos teóricos que propician el tratamiento de sucesiones indirectas, como es el tratamiento por frecuencia de las trazas extraídas de los registros de eventos. Como la Minería de Variantes y los restantes algoritmos tienen implementación en ProM se escogió este marco de trabajo para el desarrollo del algoritmo.

# CAPÍTULO 2. ALGORITMO PARA DISMINUIR LA COMPLEJIDAD TEMPORAL DE LA MINERÍA DE VARIANTES

En este capítulo se propone una solución para la disminución de la complejidad temporal de la Minería de Variantes. Este algoritmo se denominó *Variants Miner+* y está fundamentado en elementos teóricos de otros algoritmos de descubrimiento de procesos. En la concepción del *Variants Miner+* se consideraron los factores que mayor peso tienen en la complejidad algorítmica de la Minería de Variantes.

## 2.1. Descripción del algoritmo

En la complejidad algorítmica influyen directamente la cantidad de sucesiones indirectas. A partir del estudio de los algoritmos de descubrimientos se identificó como enfoque más adecuado el de analizar solamente el comportamiento más frecuente en el registro de eventos. Por esto el algoritmo diseñado tiene como objetivo limitar la cantidad de sucesiones indirectas a procesar. El factor realmente relevante no es la cantidad de sucesiones indirectas que se procesan, si no el tiempo que toma hacerlo. Por este motivo para la solución se estableció un umbral de tiempo configurable por el usuario. Este umbral va a limitar la cantidad de comportamiento a procesar durante la descomposición de cada subproceso.

Como se planteó en epígrafes anteriores, la Minería de Variantes en su primera etapa realiza un pre-procesamiento del registro de eventos durante el cual se ordenan las trazas por su frecuencia. Esto garantiza que, si se detiene el análisis de los comportamientos los que queden por procesar sean convenientemente los menos frecuentes. Por tanto, se utilizará el nuevo parámetro de configuración de tiempo para que durante la obtención de variantes de descomposición de un subproceso para un determinado patrón una vez alcanzado este umbral establecido se detenga el algoritmo.

El *Variants Miner+* va explorando en el espacio de búsqueda en función del costo de cada una de las variantes de descomposición. Esto asegura que la variante que se tiene actualmente es la solución con menor costo de las que se han generado hasta el momento. Esto garantiza que en el momento que se detenga la ejecución del algoritmo se tenga la mejor solución hasta el momento.

El costo de una solución está en términos de los comportamientos que han sido descartados como ruido o asumidos como ausencia de información. A partir de los comportamientos descartados se realiza la estimación de Aptitud y a partir de los comportamientos asumidos se estima la Precisión de la variante. Por lo tanto, es una expresión de la calidad de la propuesta de patrón que se tiene hasta ese momento. Al dejar comportamiento sin procesar se impacta en la evaluación de la descomposición para un determinado patrón y esto se ve reflejado en el perfil de Diagnóstico.

Para evitar este problema todo el comportamiento que queda por procesar una vez detenido el algoritmo debe ser analizado en función de si es coherente con la descomposición que se tiene como mejor solución. Durante este análisis se modifica la evaluación de esta descomposición en subprocesos y se añade al Perfil de Diagnóstico. A continuación, se explicará el impacto de la solución propuesta en el algoritmo de la Minería de Variantes.

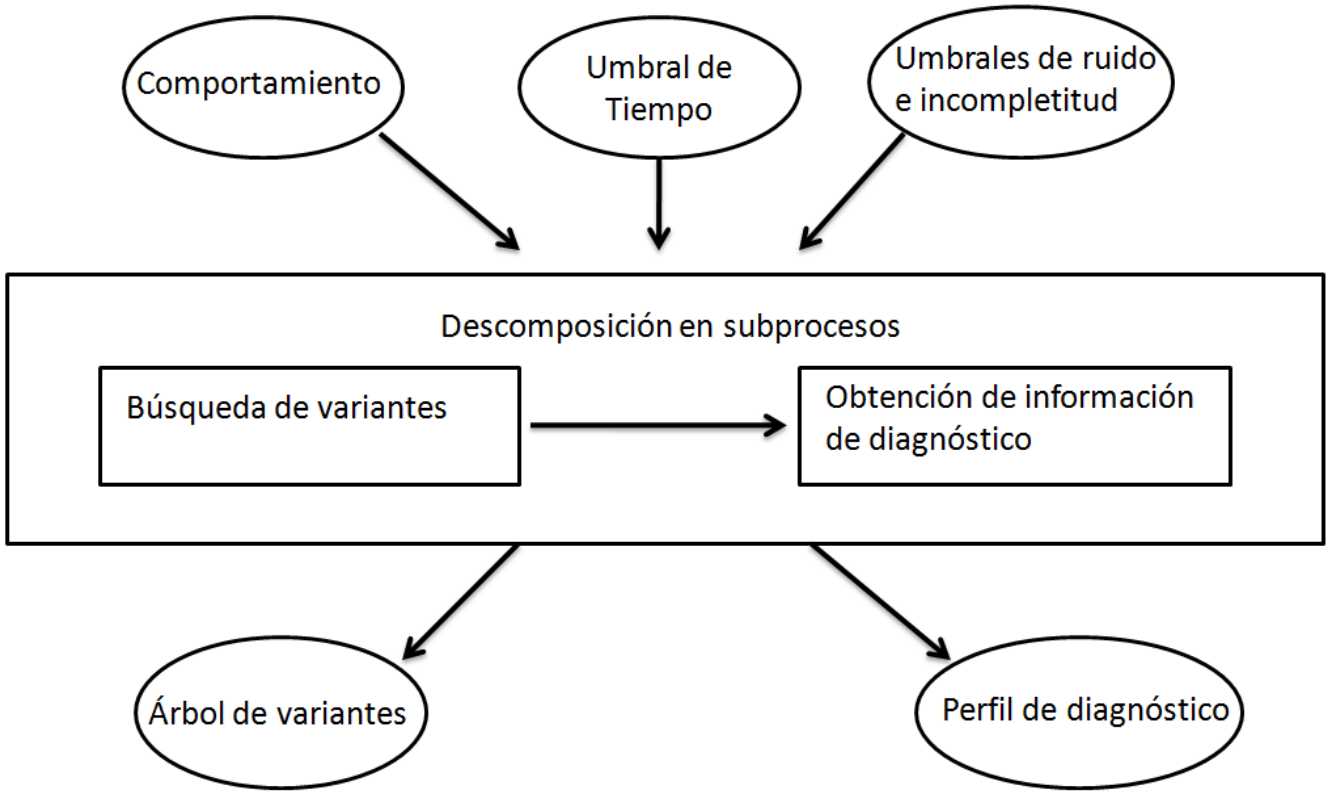


***Figura 11:*** *Componentes del algoritmo propuesto.*

## 2.2. Umbrales y Comportamiento

En la Minería de Variantes la extracción de ruido en el pre-procesamiento se ha abordado mediante la identificación de trazas anómalas. La identificación se realiza utilizando un umbral configurable por el usuario, con base en la frecuencia de las trazas en el registro de eventos (Bezerra y Wainer 2012). Los resultados dependen fuertemente de la distribución de frecuencia de las trazas, por lo que pueden ser descartadas trazas sin ruido. Además, una misma traza puede contener comportamiento que es ruido y comportamiento que no lo es.

La principal limitación en el tratamiento del ruido a través del pre-procesamiento es que, aunque se puedan identificar las trazas o incluso los comportamientos poco frecuentes no se puede determinar si estos son representativos del comportamiento típico del proceso. Discernir esto último depende de lo que se considere comportamiento típico del proceso. Una aproximación a este comportamiento es

el que expresa el modelo descubierto. Por tanto, es preferible tratar el ruido durante la construcción del modelo del proceso (Pérez-Alfonso 2015). Para esto la Minería de Variantes incluye además umbrales de ruido para cada uno de los patrones de control de flujo.

En dependencia de los eventos ausentes y la asunción de completitud de cada técnica de minería de procesos los registros de eventos pueden considerarse incompletos o no. La ausencia de eventos en las trazas es una manifestación de dos problemas de calidad de los registros de eventos: casos ausentes y eventos ausentes (Bose, Mans y van der Aalst, W.M.P. 2013). El primer problema se refiere a casos cuya ejecución no aparece en absoluto en el registro de eventos. Un caso particular del segundo problema son las trazas parciales, las cuales pertenecen a casos en los cuales algunos eventos están fuera del rango temporal seleccionado para el registro de eventos.

Las técnicas de minería de procesos deben asumir que no todo el comportamiento está en el registro de eventos, debido a la baja probabilidad de que exista completitud total. Los algoritmos que tienen restricciones poco severas respecto a la completitud del registro de eventos tienen más éxito en el tratamiento de situaciones de ausencia de eventos en las trazas (Pérez-Alfonso 2015). Para este problema de incompletitud en los registros de eventos la Minería de Variantes tiene al igual que para el ruido un umbral de completitud para el tratamiento de la ausencia de información en el pre-procesamiento y umbrales independientes por cada patrón de control de flujo para tratarla durante la construcción del modelo del proceso.

Luego que el registro de eventos es importado por la Minería de Variantes se codifican los eventos asignándole una letra a cada uno. De esta manera se elimina la información semántica de las etiquetas de los eventos disminuyendo así la carga cognitiva durante la comprensión del proceso. Posterior a la codificación se unifican las trazas cuyas secuencias coinciden, disminuyendo el número de trazas a procesar y agrupando los comportamientos coincidentes. Las secuencias resultantes de la agrupación son ordenadas descendentemente, por su frecuencia de aparición, para expresar su importancia relativa dentro del proceso. Por último, se extraen características generales presentes en el registro de eventos, como la cantidad de trazas diferentes y la cantidad de eventos diferentes (Pérez-Alfonso 2015).

A estas trazas ya ordenadas se le extrae el comportamiento (Definición 8). Por cada comportamiento extraído se obtiene la frecuencia de su aparición en las trazas codificadas. Esta frecuencia se utiliza para establecer cuán representativo es un comportamiento del proceso que lo genera. Estos comportamientos extraídos son la base para identificar los operadores de control de flujo que deben ser utilizados para la descomposición de cada subproceso (Pérez-Alfonso 2015).

## 2.3. Descomposición en subprocesos

Para la descomposición en subprocesos se utilizan como entradas el comportamiento y los umbrales establecidos por el usuario para el ruido y la ausencia de información. A partir de estas entradas se identifican las variantes de descomposición que se presentan en el árbol de Variantes. Para la obtención de estas variantes se ejecuta iterativamente el algoritmo de descomposición en subprocesos (Pérez-Alfonso et al. 2015).

El algoritmo a continuación descrito se incluye en el presente trabajo porque en la línea 11 se llama al algoritmo de búsqueda de variantes y en la línea 19 al de obtención de información de diagnóstico. Este algoritmo es tomado la Minería de Variantes (Pérez-Alfonso 2015) y se mantienen las entradas y salidas para mantener coherencia con el resto del método.

**Algoritmo 1:**

**Entrada**: registro de eventos (L), umbrales de ruido y completitud (T)

**Salida**: Árbol Variantes (VT), Perfil de Diagnóstico (D)

1: **Procedimiento** Decompose(L, T)

2: W ← {→, ∧, ×, ∨, ⟳} « Operadores de control de flujo para la descomposición

3: $sn_1$ ← CreateSubprocess(L) « Crear un nodo subproceso relativo a P

4: V T ← $sn_1$ « Asignar el nodo $sn_1$ como raíz de V T

5: S' ← {$sn_1$} « Crear una cola para los subprocesos pendientes de análisis.

6: **while** |S'| > 0 **do**

7: $sn_i$ ← GetFirst(S') « Extraer el primer nodo subproceso en S

8: **if** $sn_i$ contiene más de una actividad **then**

9: $\beta_i$ ← ExtractBehavior($sn_i$) «Extraer el comportamiento contenido en $l_i$

10: **for all** w en W **do**

11: V ← FindVariants(w, $\beta_i$, T ) «Algoritmo 2

12: **if** |V| > 0 **then** . « Si $s_i$ puede ser descompuesto por *w* dentro de T

13: $pn_i^w$ ← CreateOperatorNode(w, V)

14: **forall** nodo subproceso k en V do

15: adicionar el nodo subproceso k como hijo de $pn_i^w$ «Modificar VT

16: adicionar el nodo subproceso k a S'

17: **endfor**

18: adicionar el nodo $pn_{\mathrm{i}}^{w}$ como hijo del nodo $sn_i$ «Modificar VT

19: $d_{\mathrm{i}}^{w} \leftarrow$ CreateDiagnosisInformation (w, $\beta_i$ , V, L) « Algoritmo 3

20: D ← D + $d_{\mathrm{i}}^{w}$ « Adicionar $d_{\mathrm{i}}^{w}$ al Perfil de Diagnóstico

21: **end if**

22: **end for**

23: **end if**

24: **end while**

25: **return** VT, D

26: **end Procedimiento**

Siendo $S$ el conjunto de todos los subprocesos del proceso $P$, para cada subproceso $s_i \in S$ se busca una variante de descomposición por cada patrón de control de flujo. Al registro de eventos $L$ se le extrae un sublog $l_i$ por cada subproceso $s_i$ conteniendo solamente las trazas que tienen los eventos de las actividades representadas por $s_i$. Para determinar las variantes de descomposición de $s_i$, se utiliza el comportamiento extraído del sublog $l_i$. Al comportamiento del proceso en $s_i$ para el patrón que representa el operador $w$ y que no forma parte de $\beta_i$ se le denota $\vartheta_{\mathrm{i}}^{w}$. Siempre que $\beta_i \setminus \beta_{\mathrm{i}}^{w}$ pueda ser descartado considerando el umbral de ruido y $\vartheta_{\mathrm{i}}^{w}$ pueda ser incluido dentro del umbral de completitud se puede encontrar una variante de descomposición para el subproceso $s_i$ (Pérez-Alfonso 2015).

A la cola $S'$ se añaden los subprocesos que se generan por cada descomposición para un posterior análisis. Un subproceso $s_i$ es analizado solo si incluye más de una actividad. A medida que se identifican las variantes se va construyendo el árbol (VT) al añadir como hijo del subproceso al que corresponde la variante al nodo operador y a su vez los nuevos nodos subprocesos resultantes de la descomposición como hijos de este (Pérez-Alfonso 2015).

### 2.3.1. Algoritmo de búsqueda de variantes

En la línea 11 del algoritmo de descomposición en subprocesos se hace la llamada a la búsqueda de variantes la cual se describe a continuación. El objetivo de este algoritmo es encontrar una posible descomposición $\gamma_{\mathrm{i}}^{w}$ para el subproceso $s_i$ dado el operador de control de flujo $w$ que contiene $n$ conjuntos disjuntos de actividades pertenecientes al sublog del subproceso. El algoritmo de búsqueda

de variantes busca una descomposición con estas características minimizando el comportamiento descartado y el comportamiento asumido. Cada elemento de comportamiento $b \in \beta_i$ en esta búsqueda es procesado en orden descendente con respecto a su frecuencia en las trazas de ejecución (Pérez-Alfonso 2015).

La búsqueda de variantes se basa en la Búsqueda de costo uniforme (Pérez-Alfonso 2015). Este algoritmo de búsqueda expande el nodo con menor costo de camino. Si para cada paso el costo es igual o mayor que una constante ε queda garantizado que el método es completo y óptimo. En el peor caso de este método de búsqueda la complejidad espacial y temporal puede ser definida por la ecuación $r^{1+C/\varepsilon}$ donde $C$ es el costo del camino de la solución óptima y $r$ es el factor de ramificación del árbol. Si todos los pasos de un nodo a otro tienen costos iguales la complejidad es idéntica a una búsqueda a lo ancho (Russell et al. 1995).

En este algoritmo cada nodo en el espacio de búsqueda $\psi_n$ está definido por $\gamma_i^w$, el comportamiento no procesado ($\beta_i' \subseteq \beta_i$), el comportamiento descartado ($\eta_i^w$) y el comportamiento asumido $\theta_i^w$. Cuando se añaden nuevos nodos si existen algunos con la misma descomposición permanece el que tiene menor cantidad de procesamiento por procesar independientemente del costo del camino. Se alcanza el objetivo cuando $\beta_i'$ ha sido procesado y $\gamma_i^w$ tiene al menos dos conjuntos disjuntos (Pérez-Alfonso 2015).

**Algoritmo 2:**

**Entradas:** operador patrón de control de flujo(w), comportamiento del proceso$l_i\left(\beta_{l_i}\right)$, umbrales de ruido y completitud (T), umbral de tiempo configurable por el usuario(X)

**Salidas:** variante de descomposición en subproceso según el patrón de control de flujo w

**1**: **procedure** **FINDVARIANTS**($w$,$\beta_i$,T)

**2:** $X_i$←StartCountingTime(0) «Se inicializa el cronómetro

**3**: $\beta_i^{'}$← **SORTBYFREQUENCY**($\beta_i$) «Ordena comportamiento en orden descendente

**4**: $T_w$← $T[w]$ «Umbrales para el patrón $w$

**5**: $\gamma_i^w$ ← **CREATEFIRSTDECOMPOSITIONPROPOSAL**($w$,$\beta_i^{'}$, $T_w$)

**6**: **if** $\gamma_i^w = \emptyset$ **then** «$\beta_i^{'}$no es suficiente para crear una propuesta bajo $T_w$

**7**: **return**∅

**8**: **end if**

**9**: $\psi_1$ ←**CREATESEARCHNODE**($\beta_i^{'}$, $\gamma_i^w$)

**10**: Ψ←{$\psi_1$} «Cola de nodos abiertos ordenada por prioridad según el costo del camino

**11**: **while** $\psi_n \leftarrow$ extraer primer nodo en $\Psi$ **do**

**11**: **if** $\psi_n$ es "meta" & $\beta_i^{'} = \emptyset$ **then**

**12**: $g \leftarrow \psi_n$

**13**: **else if** $\psi_n$ no ha sido visitado **then**

**14**: $\psi_n \leftarrow$ **SEARCHNEIGHBORS**($w$, $\psi_n$, $\beta_i^{'}$, $T_w$)

**15**: $\Psi \leftarrow \Psi + \Psi_n$ «Adicionar vecinos como nodos sin visitar

**16**: **end if**

**17**: **if** $X_i$ > X «Si el tiempo transcurrido es mayor que el umbral definido

**18:** SP ← **CREATENEWSUBPROCESSES**(w, $\gamma_i^w$, $\beta_i$) «Crear los subprocesos

**19:** **returnCREATEVARIANT**(SP, $\psi_n$, $\beta_i$) «Devolver mejor variante hasta el momento

**20:** **endif**

**21**: **end while**

**22**: **if** $g \neq null$ **then** «Si se encontró una variante de descomposición

**23**: $\gamma_i^w \leftarrow$ descomposición potencial contenida en el nodo $g$

**24:** SP ← **CREATENEWSUBPROCESSES**(w, $\gamma_i^w$, $\beta_i$) «Crear los subprocesos

**25**: **returnCREATEVARIANT**(SP, g, $\beta_i$) «Crear la variante identificada

**26**: **else**

**27**: **return** $\emptyset$

**28**: **end if**

**29**: **end procedure**

Como parte de las modificaciones realizadas en este algoritmo se insertó en la línea 2 la inicialización de un contador de tiempo. Este contador permite contar el tiempo que transcurre desde el inicio de la búsqueda de variantes hasta el momento en que pasa el tiempo definido por el umbral del usuario.

De las líneas 3 a la 16 se mantiene igual que la Minería de Variantes. **Línea 3:** se ordena el comportamiento en orden descendente según las frecuencias. **Línea 4:** se extraen los umbrales de ruido e incompletitud para el patrón que se está analizando. **Línea 5:** es propuesta la primera descomposición en subprocesos teniendo en cuenta el patrón, sus umbrales y el comportamiento ordenado. **Líneas 6-7:** si el comportamiento correspondiente a este subproceso no es suficiente para crear una propuesta bajo el patrón analizado se detiene el algoritmo. **Línea 8:** en caso contrario se crea el primer nodo de búsqueda. **Línea 9:** partiendo del primer nodo se crea una cola por prioridad

según el costo del camino para los nodos abiertos. **Línea 10-12:** se itera sobre la cola de nodos de búsqueda, si para el nodo extraído ya se procesó todo el comportamiento del *sublog* significa que es un nodo meta. **Líneas 13-16:** En caso contrario si el nodo aún no ha sido visitado se hace una búsqueda de sus vecinos y se añaden a la cola con prioridad. Para esta búsqueda se tiene en cuenta el patrón de control de flujo, sus umbrales, el nodo de búsqueda actual y el comportamiento. Los vecinos de un nodo son generados por el comportamiento en función de si este es coherente o no con esa descomposición. En caso de que un comportamiento no esté en correspondencia con una descomposición se generan sus vecinos. Los vecinos de una descomposición dado un comportamiento incoherente son el mismo nodo, pero asumiendo ruido o se considera que hay ausencia de información y se genera otra descomposición que incluya este comportamiento incoherente.

Las **líneas 17-19** contienen otra modificación. Si el tiempo de ejecución del algoritmo sobrepasa al valor del umbral de tiempo insertado por el usuario se crea una nueva descomposición con la mejor variante hasta el momento. Se consideran el patrón de control de flujo y el comportamiento. Luego se devuelve esta propuesta y se detiene el algoritmo.

Durante el resto del algoritmo no se realizó ningún otro cambio. En las **líneas 22-25** si se encontró un nodo meta se crea una nueva descomposición con la información contenida en este nodo, el patrón y el comportamiento. Luego se devuelve esta variante de descomposición y se detiene el algoritmo. **Líneas 26-27** en caso de no encontrar un nodo meta se detiene el algoritmo y se devuelve ∅.

La mayor parte del comportamiento está compuesto por sucesiones indirectas. Por tanto, al detener el algoritmo en un tiempo determinado se limita la cantidad de estas sucesiones. Esto trae consigo un impacto positivo en la complejidad del algoritmo ya que se está incidiendo directamente sobre el número de Bell. Pero dejar comportamiento sin procesar trae consecuencias para la evaluación de la variante de descomposición. Por tanto, este comportamiento sin procesar es necesario considerarlo en la información de diagnóstico. En el siguiente epígrafe se explicará cómo se maneja este problema durante la extracción de la información de diagnóstico.

### 2.3.2. Obtención de información de diagnóstico

Luego de crear la descomposición en subprocesos como se describió en el algoritmo anterior, se vuelve al algoritmo 1. En la línea 19 se hace llamada a la obtención del perfil de diagnóstico teniendo como entrada el comportamiento la variante de descomposición y el registro de eventos. El Perfil de Diagnóstico (Definición 9) está asociado al Árbol de Variantes, se construye a la vez que se obtienen las variantes de descomposición. Para ellos se extrae de cada variante identificada durante la

descomposición su información de diagnóstico $d_i^w$ y se adiciona al Perfil. En este algoritmo se describe la obtención de $d_i^w$ y se adiciona al Perfil de Diagnóstico (Pérez-Alfonso 2015).

El algoritmo 3 recorre el árbol de búsqueda generado y recopila los comportamientos descartados como ruido y los comportamientos asumidos como ausencia de información en la identificación de patrones de cada variante. Para cada variante se le estima la aptitud a partir de los comportamientos descartados y la Precisión a partir de los comportamientos asumidos (Pérez-Alfonso 2015).

Como se describió con anterioridad es necesario tener en cuenta los comportamientos restantes una vez se interrumpa la ejecución del algoritmo. Si se detuvo la ejecución del algoritmo 2 debido al umbral de tiempo los comportamientos que en ese momento quedan por procesar se guardan en el nodo para su análisis en este algoritmo. Para esto se propone la siguiente modificación para la obtención de información de diagnóstico. Se adiciona un nuevo método para analizar si el nodo meta tiene comportamiento restante por procesar.

Para el análisis del comportamiento restante se tiene en cuenta además el patrón de control de flujo al que pertenece la variante de descomposición en subprocesos que se está analizando en el momento. Cada comportamiento se analiza y si no es coherente con el nodo meta se descarta como ruido o se considera como ausencia de información, luego se actualiza con esta información la lista correspondiente. Para determinar si el comportamiento está acorde a la descomposición se utilizan las siguientes reglas por cada patrón.

**Secuencia:** Su objetivo es determinar si el subproceso se puede descomponer en subprocesos ordenados secuencialmente. Para ello se procesan las sucesiones directas. Existe una descomposición mediante secuencia si para cualquier sucesión directa donde la primera actividad está en un subproceso entonces la segunda está en ese mismo o en el siguiente y si además no existe sucesión indirecta entre la segunda actividad y la primera. Una vez analizadas todas las sucesiones directas e indirectas se procesan los eventos que inician y finalizan trazas. Para los primeros se verifica que estén en el primer subproceso de descomposición y para los segundos que estén en el último (Pérez-Alfonso 2015).

**Lazo:** En este patrón se verifica si el subproceso puede descomponerse en otros dos, uno llamado **Do** y otro **Redo**. Los comportamientos procesados para esto son las sucesiones directas, los eventos que inician y finalizan trazas y los repetidos. Existe una descomposición por lazo si todas las actividades cuyos eventos inician o finalizan trazas forman parte del **Do**. Para toda sucesión directa entre **a** y **b**: si **a** pertenece al **Do** y **b** al **Redo**, **a** debe iniciar al menos una traza y si **a** pertenece al **Redo** y **b** al **Do**, **b** debe concluir al menos una traza (Pérez-Alfonso 2015).

**Paralelismo:** Para determinar si puede descomponerse el subproceso en otros que se ejecuten concurrentemente se procesan las sucesiones indirectas. Existe una descomposición por paralelismo si existe una sucesiones indirectas **ab** y **ba** y **a** y **b** están en diferentes subprocesos y si en cada una de las trazas aparece al menos un evento de los subprocesos concurrentes (Pérez-Alfonso 2015).

**Selección Exclusiva:** Para que exista una ejecución excluyente de los subprocesos si para cada sucesión indirecta **ab**, **a** está en el mismo subproceso que **b** (Pérez-Alfonso 2015).

**Selección no Exclusiva:** Para determinar una variante de descomposición en subprocesos que se ejecuten en selección no exclusiva debe existir para toda actividad **a** y **b** que se encuentren en diferentes subprocesos debe existir al menos una sucesión indirecta **ab** y otra **ba**. Además en al menos otra traza no existe representación de todos los subprocesos (Pérez-Alfonso 2015).

**Algoritmo 3:**

**Entradas:** operador (w), comportamiento $(\beta_{l_i})$, variante (V), registro de eventos (L)

**Salidas:** información de diagnóstico $d_i^w$ asociada a la variante V

**1**: **Procedimiento** CreateDiagnosisInformation(w, $\beta_i$ , V, L)

**2**: g← extraer el nodo "meta" en V

**3**: $l_i$← ExtractSublog(V, L). «Extraer la sección del registro de eventos

**4**: $n_i^w \leftarrow \{\}$ «Crear el conjunto de comportamientos descartados

**5**: $\theta_i^w \leftarrow \{\}$ «Crear el conjunto de comportamientos asumidos

**6**: $\psi_1$← g

**7:** $R$← $\psi_1$ «Extraer comportamiento por procesar del nodo meta

**8:** **while** $R_k$← extraer primer nodo en $R$ **do** «Si queda comportamiento por procesar

**9:** **if** DiscardedBehavior($R_k$, $\beta_i$, w) «Si el comportamiento se descarta

**10:** $n_i^w \leftarrow R_k$ «Adicionar el comportamiento descartado

**11:** **end if**

**12:** **if** TakenBehavior($R_k$, $\beta_i$, w) «Si el comportamiento se asume

**13:** $\theta_i^w \leftarrow R_k$ «Adicionar el comportamiento asumido

**14:** **end if**

**15**: **while** $\psi_1$ no es nulo **do**

**16**: $n_i^w \leftarrow n_i^w + (\psi_1 \rightarrow n_i^w)$ «Adicionar el comportamiento descartado en $\psi_1$

**17**: $\theta_i^w \leftarrow \theta_i^w + (\psi_1 \rightarrow \theta_i^w)$ «Adicionar el comportamiento asumido en $\psi_1$

**18**: $\psi_1 \leftarrow$GetParent($\psi_1$) «Moverse hacia el nodo padre

**19**: **endwhile**

**20**: f←CalculateFitnessEstimation($\mathrm{n}_\mathrm{i}^w$, $\beta_\mathrm{i}$)

**21**: ρ←CalculatePrecisionEstimation($\theta_\mathrm{i}^w$, w, V)

**22**: $\mathrm{d}_\mathrm{i}^w \leftarrow [l_i, \mathrm{n}_\mathrm{i}^w, \theta_\mathrm{i}^w, \mathrm{f}, \mathrm{p}]$

**23**: **return**$\mathrm{d}_\mathrm{i}^w$

**24**: **end Procedimiento**

A continuación, una descripción detallada del algoritmo propuesto para la obtención de información de diagnóstico. **Línea 2:** se extrae el nodo meta contenido en la variante de descomposición en subprocesos. **Línea 3:** a partir del registro de eventos se extrae el *sublog* correspondiente a la variante de descomposición. **Líneas 4-5:** se crean las listas de comportamientos descartados y asumidos respectivamente. **Línea 6:** Se extrae el último nodo de búsqueda de la variante de descomposición en subprocesos. Hasta aquí el algoritmo no contiene cambios respecto a la Minería de Variantes.

De las líneas 7 a la 14 se realizó la modificación para tratar el comportamiento que quedó por procesar si la búsqueda fue interrumpida debido al límite de tiempo. Destacar que esta lista existe sólo si se ejecutaron las líneas 17-19 del algoritmo 2. **Línea 7:** se extrae el comportamiento que quedó por procesar para este último nodo. **Línea 8:** iterar por los comportamientos restantes a procesar mientras la lista no sea vacía. **Línea 9:** se ejecuta un método que a partir del patrón y el comportamiento y teniendo en cuenta las reglas descritas de cada patrón explicadas anteriormente se determina si este comportamiento en cuestión debe ser descartado como ruido. **Línea 10:** si se determinó que el comportamiento se debe descartar se añade a la lista correspondiente. **Líneas 12-13:** al igual que el anterior se ejecuta un método con las mismas entradas, pero esta vez para determinar si el comportamiento debe ser asumido como ausencia de información.

A partir de aquí el algoritmo continúa al igual que el original de la Minería de Variantes. **Línea 15:** partiendo del último nodo se itera hacia atrás en el árbol de variantes para actualizar la información de diagnóstico. **Líneas 16-17:** los comportamientos descartados como ruido o asumidos como ausencia de información del actual nodo de búsqueda se adicionan a la lista correspondiente. **Línea 18:** se mueve el análisis hacia el padre del nodo de búsqueda actual. **Línea 20:** a partir del contenido de la lista de comportamientos descartados como ruido se calcula una estimación para la aptitud de la variante de descomposición. **Línea 21:** a partir de la información almacenada en la lista de comportamientos asumidos como ausencia de información se calcula una estimación de la precisión

de la variante. **Línea 22-23:** se crea y devuelve la información de diagnóstico asociada a la variante de descomposición en subprocesos analizada.

## 2.4. Análisis de los componentes de la Minería de Variantes

El método de Minería de Variantes está implementado en el marco de trabajo ProM por el *plugin Variants Miner.* Para la implementación del *Variants Miner+* se realizó un análisis del *plugin* donde se identificaron sus componentes.

El marco de trabajo ProM soporta diferentes tipos de complementos. El primer tipo de complementos se refiere a aquellos que realizan el procesamiento necesario para la obtención de determinados objetos, a partir de objetos de entrada. El segundo tipo de complementos se utiliza para realizar la representación de los objetos, lo cual posibilita su visualización por parte de los usuarios. El último tipo de complementos se utiliza para realizar los procesos de importación y exportación de recursos en el marco de trabajo. No se realizó ninguna modificación en el complemento de visualización, ni en el de importación y exportación de recursos. En el primer tipo de complemento de la Minería de Variantes se hicieron modificaciones para realizar el procesamiento necesario para obtener el Árbol de Variantes a partir de un registro de eventos. En la Figura 12 se muestra la estructura correspondiente al complemento que posibilita la obtención del Árbol de Variantes.

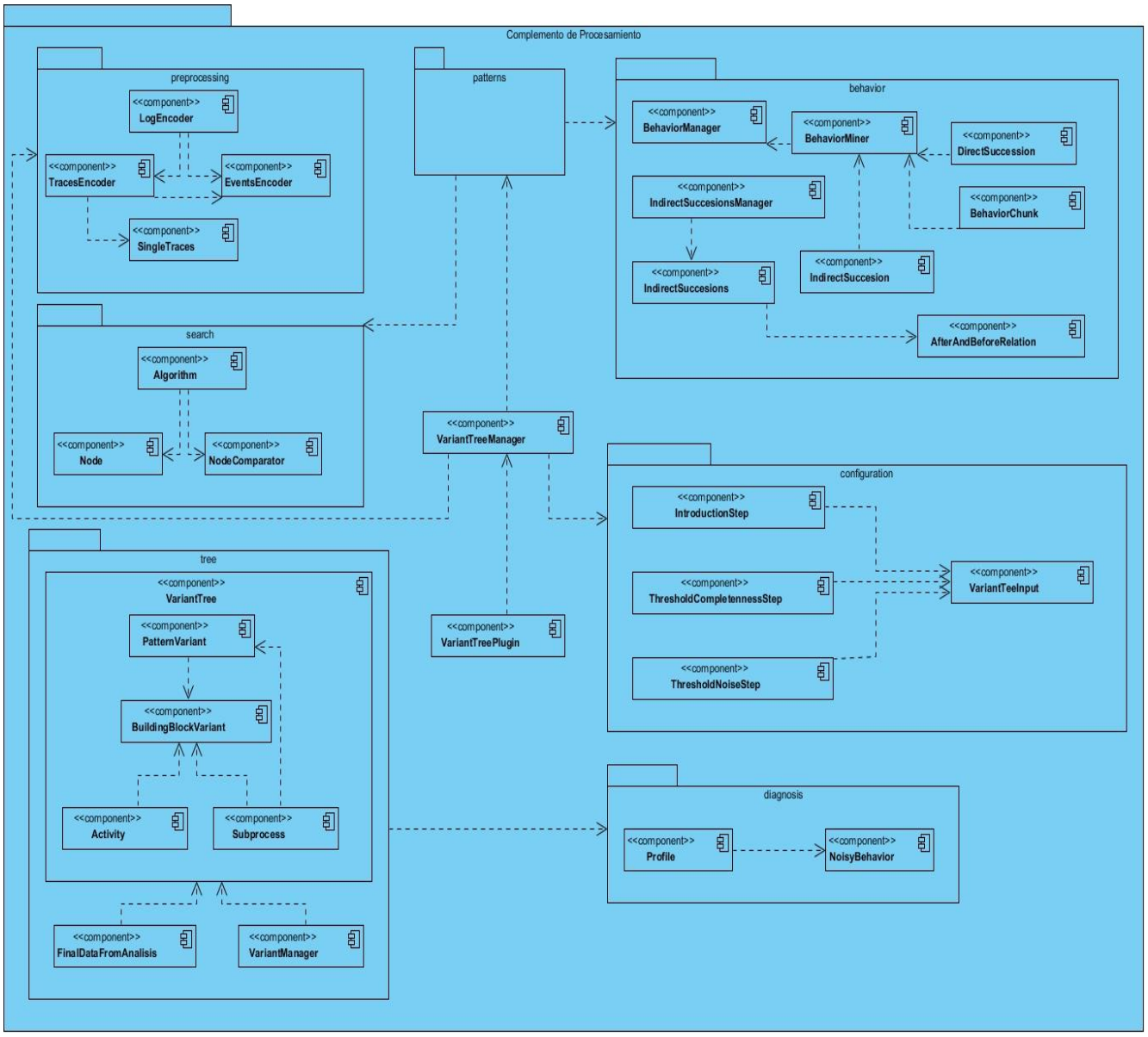


***Figura 12:*** *Estructura del complemento del procesamiento* (Pupo-Hernández y López-Jiménez 2014).

Este complemento está estructurado en 7 paquetes: *configuration*, *preprocessing*, *tree*, *diagnosis*, *behavior*, *search* y *patterns*. Cada uno de estos paquetes contiene los componentes necesarios para la realización de funciones específicas dentro del complemento.

En el paquete *configuration* se encuentran los componentes encargados de obtener y almacenar los parámetros de configuración introducidos por el usuario. Los datos obtenidos son empleados durante la identificación de los patrones de control de flujo. El paquete *preprocessing* contiene los componentes encargados de realizar la codificación de las actividades presentes en el registro de eventos, extraer los datos generales del registro de eventos y agrupar las trazas iguales. Los componentes pertenecientes a la estructura del Árbol de Variantes se encuentran ubicados en el

paquete *tree*. En el paquete *diagnosis* se ubican los componentes asociados al perfil de diagnóstico. Los componentes presentes en el paquete *behavior* realizan la extracción del comportamiento relevante dentro del proceso. Dentro del paquete *search* se encuentran los componentes correspondientes al algoritmo de búsqueda empleado durante el descubrimiento de los patrones de control de flujo. Por último, en el paquete *patterns*, mostrado en la Figura 13, se encuentran los componentes empleados para realizar el descubrimiento de los patrones de control de flujo. Asociado al descubrimiento de cada uno de los patrones de control de flujo existe un componente especializado en la realización del procesamiento necesario (Pupo-Hernández y López-Jiménez 2014).

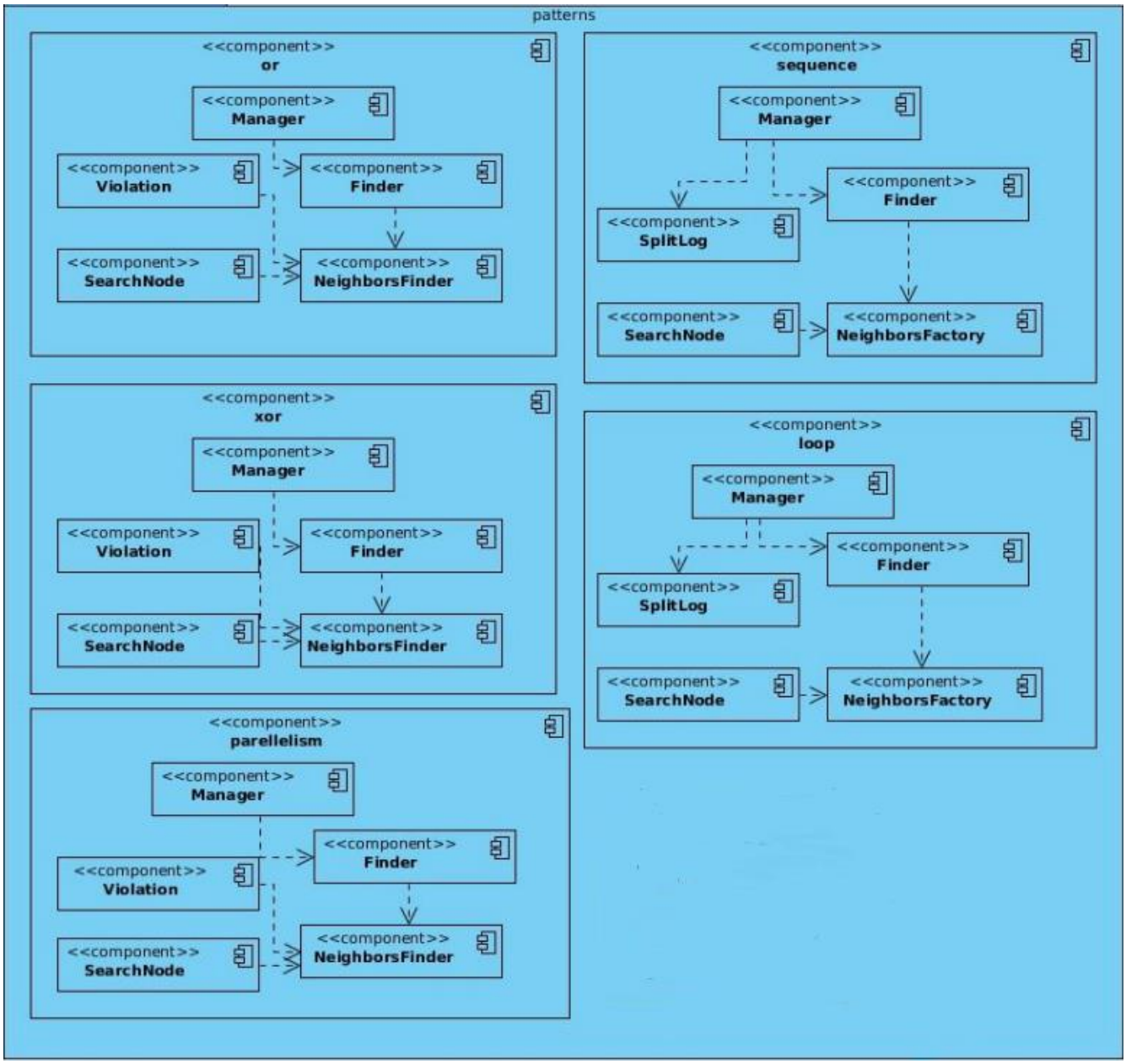


***Figura 13:*** *Componentes del paquete patterns* (Pupo-Hernández y López-Jiménez 2014).

## 2.5. Implementación del algoritmo

Para la implementación del algoritmo propuesto se realizaron modificaciones en varios de los paquetes de la Minería de Variantes. En el paquete *configuration* se insertó una nueva clase llamada *ThresholdSearchStep* la cuál añade una nueva ventana a la configuración del *plugin* para insertar el límite de tiempo que se utilizará en la búsqueda (Figura 14). El usuario tiene la opción de marcar si desea utilizar o no este umbral, de no hacerlo estaría ejecutando el *plugin* original. El umbral es mostrado en minutos, pero internamente se trabaja en milisegundos. Su rango es de 1 segundo a 100 minutos.

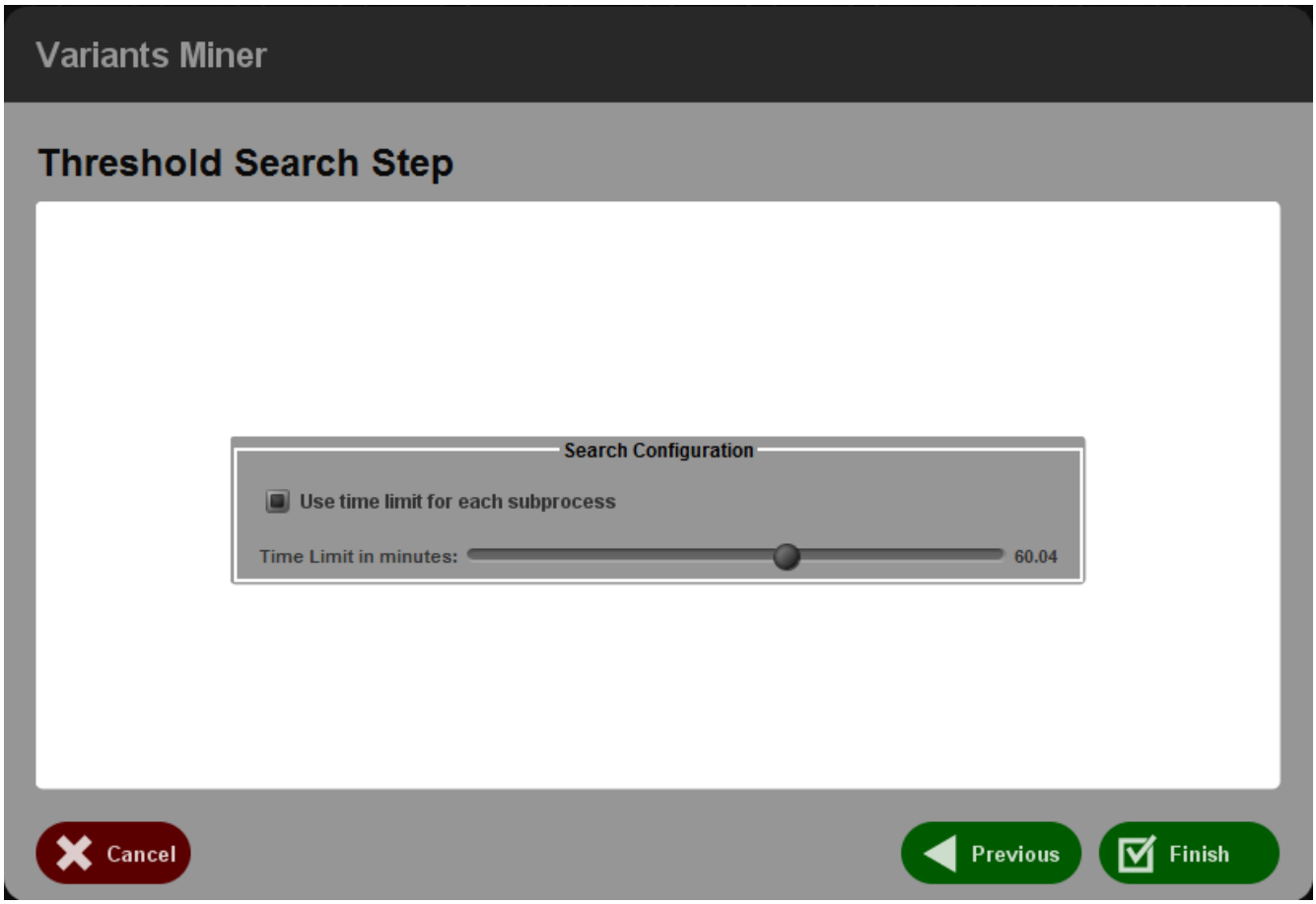


***Figura 14:*** *Ventana de configuración para insertar el límite de tiempo en el que se desea detener la descomposición en subprocesos.*

Este nuevo umbral de tiempo se adiciona a la configuración de entrada. A través del *VariantTreeManager* se lleva al paquete *Patterns* y de allí al *search*. En este último es donde se encuentra el algoritmo de búsqueda de variantes. En la clase *UniformCostAlgorithm* de este paquete se inicializa un cronómetro utilizando la clase *GregorianCallendar* del lenguaje java. Una vez este

cronómetro alcanza el tiempo límite establecido en el umbral se detiene la ejecución del algoritmo y se da paso al procesamiento del comportamiento restante.

El comportamiento que quedó sin procesar se trata independiente por cada patrón en el paquete *patterns*. Como se puede apreciar en la Figura 13 dentro de este paquete se puede agrupar los patrones en dos grupos a partir de su estructura. Todos tienen las clases *Manager, Finder* y *SearchNode,* pero la secuencia y lazo utilizan además las clases *NeighborsFactory* y *SplitLog*; mientras que la selección exclusiva, selección no exclusiva y paralelismo utilizan las clases *NeighborsFinder* y *Violation*.

Aprovechando que todos tienen la clase *SearchNode* y que en esta se encuentra los datos necesarios de los nodos de búsqueda se insertaron allí las siguientes modificaciones. Una lista como atributo llamada *remainingBehavior* y un método llamado *obtainRemainingBehavior* que le adiciona los comportamientos restantes. Este método se implementa independiente para cada uno de los patrones. Se utilizan las reglas de los patrones para determinar si los comportamientos son coherentes con la descomposición en subprocesos contenida en el nodo. Si no concuerda se añaden a la lista de comportamientos restantes.

La lista de comportamiento que quedó por procesar de cada nodo se retoma en la clase *Finder* de cada patrón. Allí se analizan los comportamientos incoherentes con la descomposición y se modifica los valores de ruido y completitud de la información de diagnóstico para luego esta añadirla al Perfil de Diagnóstico.

## 2.6. Conclusiones parciales

Durante este capítulo se presentó el *Variants Miner+* para disminuir la complejidad computacional del método Minería de Variantes, el algoritmo propuesto incluye la utilización de un umbral de tiempo configurable por el usuario y modificaciones realizadas a la descomposición en subprocesos de la Minería de Variantes, modificando sus algoritmos de búsqueda de variantes y generación del perfil de diagnóstico. A partir del diseño e implementación del *Variants Miner+* se tienen las siguientes conclusiones:

- Una vez alcanzado el umbral de tiempo adicionado para disminuir el tiempo de descomposición en subprocesos se garantiza que no se procese el resto del comportamiento que contiene el registro de eventos. Este comportamiento se ordenó en función de sus frecuencias en etapas anteriores de la Minería de Variantes, por esto se puede asegurar que el que queda por procesar es el menos frecuente y por lo tanto menos representativo que los comportamientos que ya estén considerados.

- La modificación al algoritmo de obtención de información de diagnóstico garantiza que todo el comportamiento que quedó por procesar se añada al perfil de diagnóstico. De esta forma todas las trazas del registro de eventos quedan consideradas en la evaluación de cada descomposición en subprocesos.

# CAPÍTULO 3. VALIDACIÓN DEL ALGORITMO

En el presente capítulo se realiza un análisis de la calidad en los modelos de procesos. Este está enmarcado en el chequeo de conformidad, una de las tres áreas de la Minería de Procesos. Se explican las dimensiones utilizadas para evaluar modelos en el chequeo de conformidad. Se propone la herramienta CoBeFra como marco de trabajo para la evaluación de calidad. Se escogen las métricas a utilizar para cada dimensión.

En el diseño de la validación se explica el pre-experimento a realizar. En él se propone un grupo de registros de eventos con determinados niveles de ruido y ausencia de información, así como diferentes cantidades de sucesiones indirectas y actividades. A partir de estos registros de eventos se realizan pruebas de tiempo y calidad. Los tiempos de ejecución del *Variants Miner+* se comparan con los de la Minería de Variantes. La evaluación de calidad obtenida por el algoritmo Varinats Miner+ se compara con la de la Minería de Variantes y con la de los modelos de proceso originales que dieron lugar a los registros de eventos. Los resultados obtenidos se analizan y se determina en qué medida disminuyó el tiempo de obtención de variantes de proceso y cuánta depreciación de la calidad trajo como consecuencia.

## 3.1. Calidad de los modelos de proceso

La evaluación de la calidad de un modelo de procesos se lleva a cabo por el chequeo de conformidad. El chequeo es un área de la minería de procesos que analiza la semejanza entre un modelo de procesos y un registro de eventos. El modelo puede haber sido construido a mano o puede ser el resultado de un algoritmo de descubrimiento de procesos. El chequeo de conformidad tiene como objetivo comparar el comportamiento de registros de eventos con el comportamiento contenido en los modelos de proceso. El objetivo es encontrar similitudes y discrepancias entre el comportamiento modelado y el comportamiento observado. El chequeo de conformidad es relevante para el mejoramiento de negocios y auditorías. Por ejemplo, el registro de eventos puede ser ejecutado en el modelo de procesos para encontrar anomalías que sugieren fraude o ineficiencias. Las técnicas de chequeo de conformidad pueden ser usadas además para medir la ejecución de los algoritmos de descubrimientos de procesos y reparar modelos que no están bien alineados con la realidad (van der Aalst, W.M.P. 2011).

Para medir la calidad de un registro de eventos se puede usar el ruido y la completitud, pero estos no dicen mucho de la calidad del modelo descubierto. Para determinar la calidad de un modelo resultado de una técnica de Minería de Procesos se emplean otras dimensiones: completitud, simplicidad, precisión y generalización (van der Aalst, W.M.P. 2011). La precisión y la generalización tienen una

relación inversa, además si un modelo es simple probablemente tiene baja Aptitud y carece de Precisión. Idoneidad de comportamiento es el término para denominar el balance correcto entre Generalización y Precisión e implica que el modelo exprese el comportamiento mínimo para representar lo más cercanamente posible la ejecución real del proceso. La idoneidad estructural se refiere a la capacidad del modelo para reflejar claramente el comportamiento registrado, con la mínima estructura posible. La obtención de un modelo apropiado, implica lograr su idoneidad en ambos sentidos (Rozinat et al. 2007; van der Aalst, W.M.P. et al. 2010).

Un modelo de procesos con buena completitud contiene el comportamiento observado en el registro de eventos. Un modelo tiene una completitud perfecta si todas las trazas en el registro pueden ser ejecutadas por el modelo de principio a fin. La completitud puede ser definida a nivel de caso para que la fracción de trazas en el registro de eventos pueda ejecutarse completamente, puede ser definida también al nivel del evento de forma que la fracción de eventos en el registro de eventos sea posible de acuerdo al modelo. Cuando se define la completitud se debe hacer muchas decisiones de diseño; como establecer la penalización si un paso necesita ser saltado y si quedan *tokens* en la red de flujo de trabajo después de la ejecución del modelo (van der Aalst, W.M.P. 2011).

La complejidad del modelo puede ser definida por el número de nodos y arcos en el grafo que representa el modelo de proceso obtenido. En el contexto de descubrimiento de procesos el modelo más simple que puede explicar el comportamiento observado en el log es el mejor modelo. La simplicidad incluye algunas métricas más sofisticadas. (van der Aalst, W.M.P. 2011).

Un modelo es preciso si no permite “mucho más” comportamiento, un modelo que no es preciso es poco ajustado (*underfitting*) o un modelo que permite comportamientos muy diferentes de los que se muestran en el registro de eventos (van der Aalst, W.M.P. 2011). Un modelo debe generalizar y no restringir comportamiento a los vistos en el registro de eventos. Un modelo que no generaliza es muy ajustado (*overfitting*). *Overfitting* es el problema en el que un modelo muy específico es generado independientemente que sea obvio que el registro de eventos solo comprende dicho comportamiento. Este tipo de modelo puede explicar un registro en particular, pero el siguiente registro de eventos puede generar un modelo de procesos completamente diferente (van der Aalst, W.M.P. 2011).

*Comprehensive Benchmarking Framework* (CoBeFra) es un marco de trabajo que contiene un set de herramientas para evaluar la calidad de un modelo de procesos. También se utiliza para evaluar el desempeño de diferentes modelos entre ellos usando múltiples métricas de chequeo de conformidad. Estas métricas están divididas en dos grupos las de exactitud y las de comprensibilidad. El grupo de exactitud contiene métricas de tres perspectivas diferentes eventos (De Weerdt, Baesens y Vanthienen 2013). Estas perspectivas son *Fitness, Precision* y *Generalization*, las cuales son equivalentes a las dimensiones que fueron explicadas anteriormente.

CoBeFra reutiliza varias bibliotecas y componentes disponibles para ProM. Puede ser iniciado como otro *plugin* de ProM y opcionalmente permite cargar un proyecto previamente guardado para resumir o reiniciar el experimento; importar y exportar se hace mediante la arquitectura proporcionada por ProM y la visualización de los resultados de las métricas fue separada en un *plugin* diferente. No obstante CoBeFra una vez iniciado descarta todas las dependencias específicas de ProM de forma que también puede ser ejecutado como una aplicación independiente (De Weerdt, Baesens y Vanthienen 2013).

## 3.2. Diseño de la validación

Para lograr una reducción del tiempo en la Minería de Variantes fue necesario limitar la cantidad de comportamiento a procesar del registro de eventos. Este comportamiento descartado puede haber sido representativo del proceso. Esto trae consigo un impacto en la calidad del modelo del proceso. Por esto se hace necesario evaluar la calidad de los modelos obtenidos y analizar si esta tuvo un deterioro considerable.

La Simplicidad considera la carga cognitiva que implican los modelos complejos producto de su densidad. Las métricas que la evalúan están basadas en la visualización y son sobre redes de *Petri*. En el caso del algoritmo propuesto se mantiene la visualización de la Minería de Variantes, por lo que se mantiene como salida un Árbol de Variantes. Por eso la evaluación de la calidad del algoritmo no se considera la dimensión de simplicidad.

Las dimensiones generalización, precisión y aptitud se refieren a la veracidad del modelo, ya que evalúan su alineación con la ejecución real del proceso (Yzquierdo-Herrera 2012). De estas fueron consideradas para la evaluación de la calidad la aptitud y precisión.

Para evaluar la calidad y el tiempo de ejecución del Variants Miner+ se realizó un pre-experimento. Se escogieron 4 registros de eventos provenientes de un total de 90 que fueron utilizados en experimentos similares (Vázquez-Barreiros, Mucientes y Lama 2015) (Pérez-Alfonso 2015). Estos registros de eventos fueron generados aleatoriamente a partir de modelos de procesos de diferente complejidad combinando todos los patrones de control de flujo. Estos rasgos determinan su idoneidad para el experimento.

En estos registros de eventos se introdujo ruido y ausencia de información realizando cuatro operaciones sobre las trazas: extracción de eventos del principio, cuerpo y final de las trazas, además de un intercambio de eventos elegidos aleatoriamente. De esta forma quedaron modificadas las trazas para diferentes porcientos: 0%, 1%, 5%, 10% y 20% (Vázquez-Barreiros, y otros, 2015). De estos grupos de registros de eventos se consideró el grupo que contiene un 20% de ruido y ausencia de información ya que estos son factores que influyen directamente en la cantidad de sucesiones

indirectas que se extraen cuando se analiza el comportamiento. Del grupo de 20% se escogieron los 4 registros de eventos con mayor cantidad de actividades ya que la cantidad de actividades también influye directamente en la complejidad algorítmica. Los datos de los registros de eventos se pueden observar en la Tabla 3.

***Tabla 3:*** *Características de los registros de eventos seleccionados para la validación.*

| Nombre | CC | CV | CA | Min EC | Max EC | Min AC | Max AC |
|---|---|---|---|---|---|---|---|
| Caminatas_700_Noisy_0.2pc.xes | 700 | 5445 | 12 | 6 | 8 | 6 | 8 |
| driveClass_700_Noisy_0.2pc.xes | 700 | 14398 | 13 | 8 | 23 | 8 | 13 |
| exampleLog_300_Noisy_0.2pc.xes | 300 | 2180 | 10 | 5 | 8 | 5 | 8 |
| groupedFollowsparallel5_700_Noisy_0.2pc.xes | 700 | 8203 | 12 | 8 | 12 | 8 | 12 |

**Leyenda:**

Nombre: El nombre del archivo que contiene el registro de eventos.

CC: Cantidad de casos.

CV: Cantidad de eventos

CA: Cantidad de actividades

MinEC: Cantidad mínima de eventos por caso

MaxEC: Cantidad máxima de eventos por caso

MinAC: Cantidad mínima de actividades por caso

MaxAC: Cantidad máxima de actividades por caso

### 3.2.1. Validación temporal

Para el análisis temporal se insertó en el código un cronómetro para medir el tiempo de ejecución del *Variants Miner+* y del algoritmo original. Para medir el tiempo se utilizó la clase *GregorianCalendar* del lenguaje de programación Java. Los tiempos se miden para ambos algoritmos utilizando los mismos registros de eventos y la misma computadora.

Los registros de eventos empleados son los propuestos en el epígrafe anterior. La computadora utilizada para las pruebas tiene las siguientes características:

Marca: HP

Modelo: Compaq 6715b

Procesador: AMD Turion 64 X2 Dual-Core Mobile Technology TL-60 (2.0-GHz, 2 x 512-KB L2 cache)

Chipset: AMD M690T Chipset

Memoria: 3072-MB 667-MHz DDR2 SDRAM

Disco Duro: 80-GB 7200 rpm SMART SATA

Sistema operativo: Windows 10 Pro 2016

Entorno de desarrollo: NetBeans IDE 8.1 RC2 (Build 201510122201)

Máquina virtual de Java: 1.7.0_71; Java HotSpot(TM) 64-Bit Server VM 24.71-b01

### 3.2.2. Validación de calidad

Para la evaluación de la calidad fueron consideradas las dimensiones de calidad Aptitud y Precisión. Para cada una de estas las métricas *AlignmentBasedFitness* (Adriansyah, Sidorova y van Dongen 2011) y *OneAlignPrecision* (Adriansyah et al. 2012) implementadas en CoBeFra (De Weerdt, Baesens y Vanthienen 2013) respectivamente. El diseño experimental para cada dimensión se resume en la Tabla 4, mediante la siguiente simbología:

G: Registros de eventos.

R: Asignación al azar.

X: Tratamiento o estímulo. X1 corresponde a la aplicación de la Minería de Variantes y X2 a la solución propuesta.

O: Observación. Evaluación de cada modelo contra el registro de eventos respectivo.

***Tabla 4:*** *Diseño experimental utilizado para cada dimensión.*

| | Modelo Original | | Árbol de Variantes | | Solución propuesta |
|---|---|---|---|---|---|
| RG1 | O1 | X1 | O2 | X2 | O3 |
| RG2 | O4 | X1 | O5 | X2 | O6 |
| RG3 | O7 | X1 | O8 | X2 | O9 |
| RG4 | O10 | X1 | O11 | X2 | O12 |

Las primeras observaciones se refieren a la evaluación de la dimensión con los modelos originales, las del segundo momento a la evaluación de los modelos obtenidos por la Minería de Variantes y la tercera a la del *Variants Miner+*. Por cada observación se ejecuta la métrica correspondiente a cada uno de los 4 registros de eventos.

Se ejecuta el *Variants Miner+* utilizando uno de estos registros de eventos y se obtiene un árbol de variantes. Del árbol se seleccionó la mejor variante utilizando la evaluación de cada una que se

extrajo en el perfil de diagnóstico. A continuación, se muestran los árboles de variantes obtenidos para cada registro de eventos.

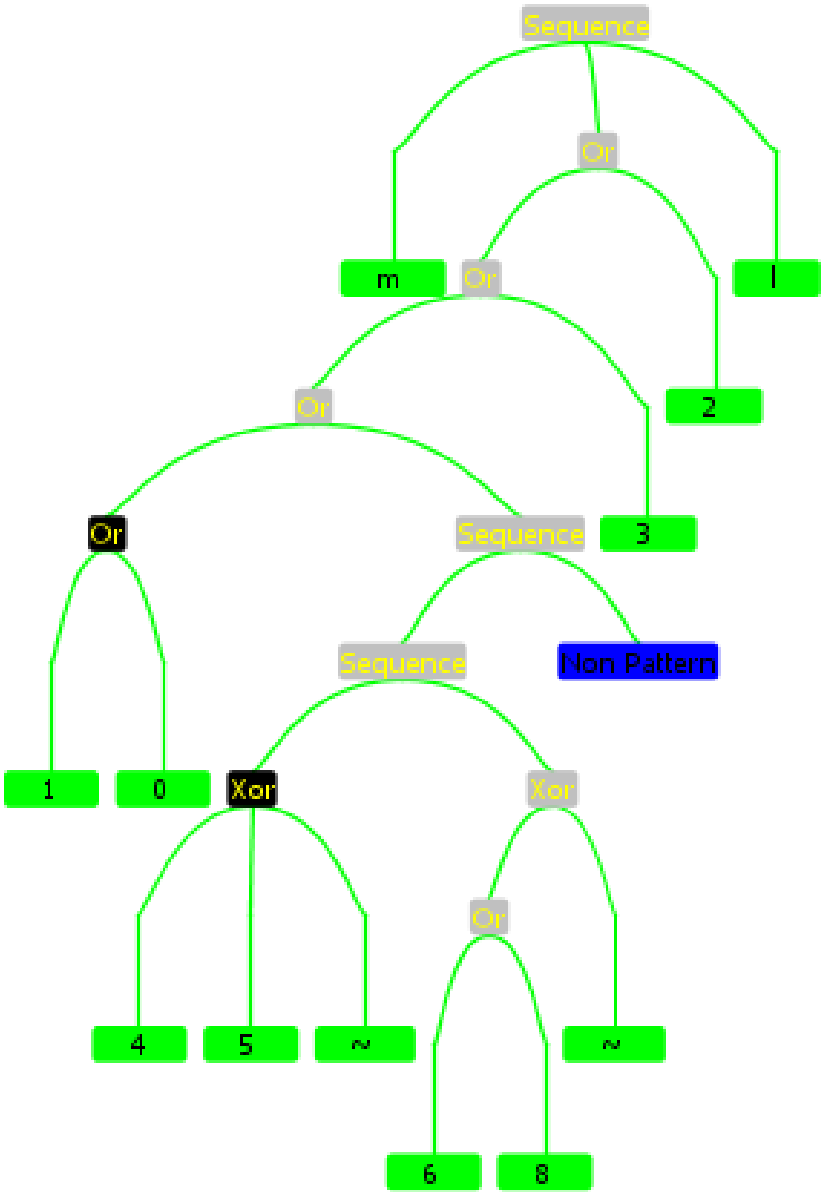


***Figura 15:*** *Visualización del árbol de variantes obtenido por el Variants Miner+ para el registro de eventos* Caminatas_700_Noisy_0.2pc.xes.

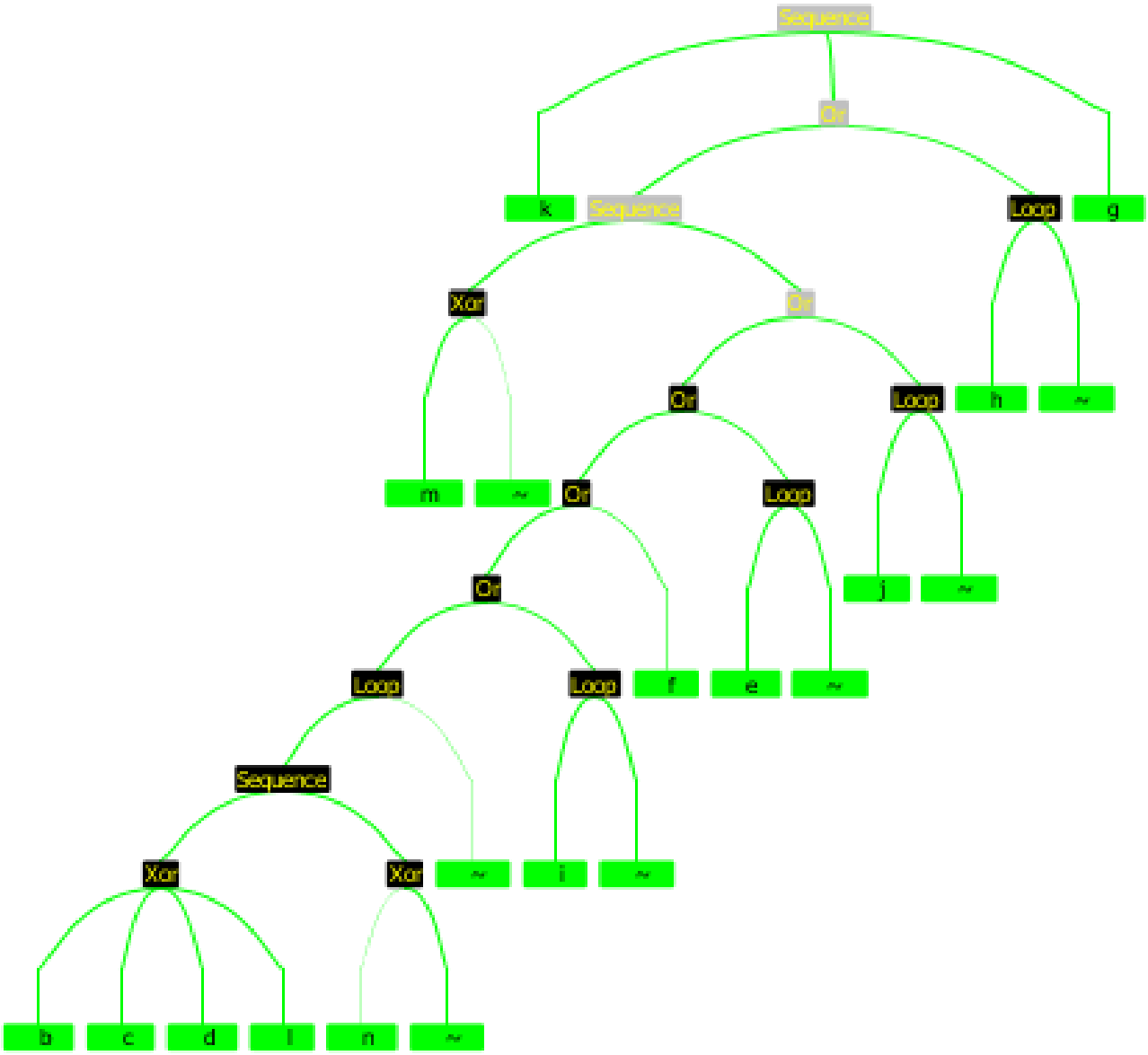


***Figura 16:*** *Visualización del árbol de variantes obtenido por el Variants Miner+ para el registro de eventos driveClass_700_Noisy_0.2pc.xes.*

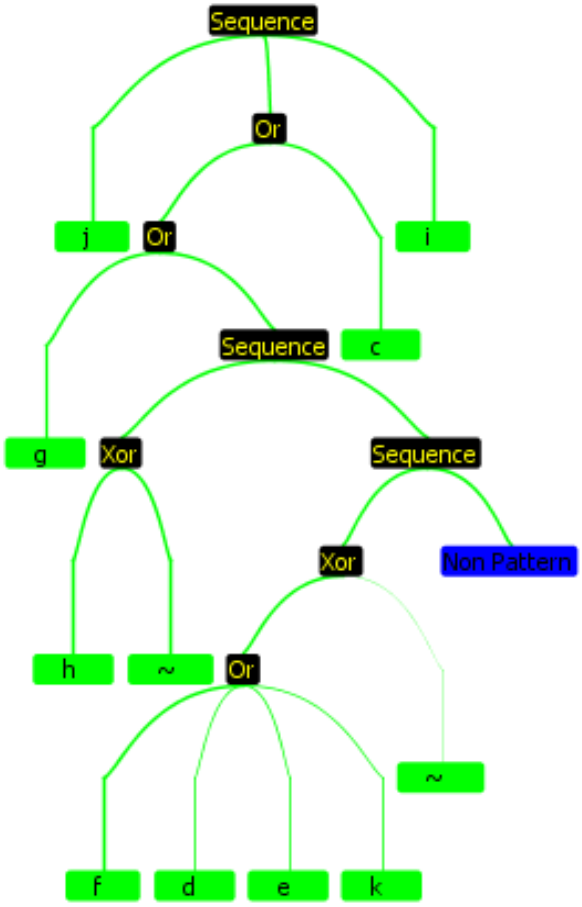


***Figura 17:*** *Visualización del árbol de variantes obtenido por el Variants Miner+ para el registro de eventos exampleLog_300_Noisy_0.2pc.xes.*

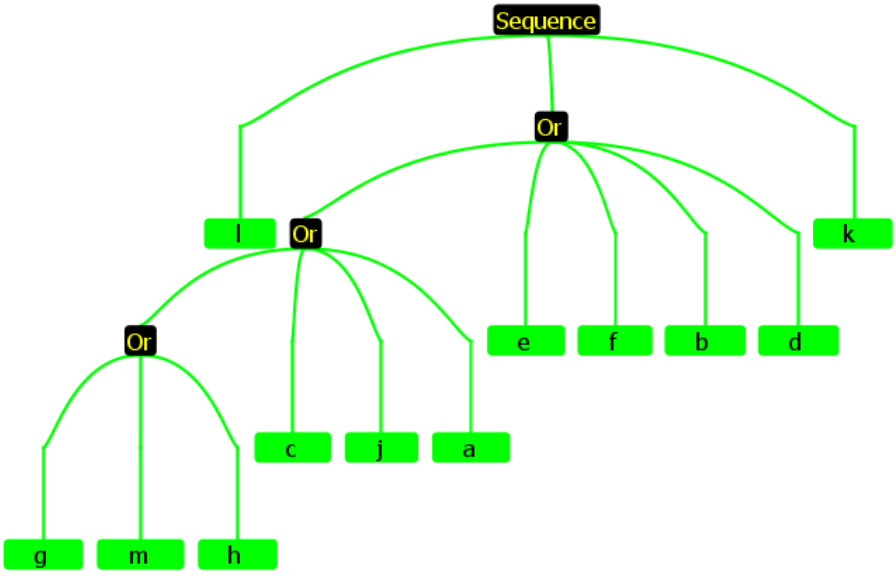


***Figura 18:*** *Visualización del árbol de variantes obtenido por el Variants Miner+ para el registro de eventos groupedFollowsparallel5_700_Noisy_0.2pc.xes.*

La variante seleccionada en cada árbol se convierte a una Red de *Petri* utilizando el *plugin VariantTree to Petri Net with Markings*. Se ejecuta el marco de trabajo CoBeFra a partir de las redes de *Petri* obtenidas, los registros de eventos y las métricas escogidas por cada dimensión. Los resultados obtenidos son una evaluación de la calidad del modelo obtenido respecto al registro de eventos en función de las dimensiones de calidad analizadas.

Los valores obtenidos representan la calidad de cada uno de los modelos con respecto al registro de eventos. Estos resultados pueden compararse con los obtenidos por (Pérez-Alfonso 2015) en su evaluación para el algoritmo de Minería de Variantes. De la misma manera pueden comparar con los resultados obtenidos por los modelos originales.

## 3.3. Análisis de los resultados

Las pruebas de tiempo fueron realizadas en ambos algoritmos. Se utilizaron los 4 registros de eventos seleccionados y la misma computadora. Se configuró para cada uno de los registros de eventos un límite de tiempo de 1 segundo por patrón de descomposición. Lo cual significa que una vez alcanzado este límite de tiempo se va a detener la obtención de variantes de descomposición para ese patrón. Alcanzado ese momento se devuelve la mejor variante obtenida hasta el momento. Entonces, se procede a la modificación de la información de diagnóstico para esa descomposición, donde se considera el comportamiento que quedó por procesar. Los resultados de tiempo obtenidos para la ejecución del algoritmo para cada uno de los registros de eventos se muestran en la Tabla 5:

***Tabla 5:*** Resultados de las pruebas de tiempo.

| Registros de eventos | Minería de Variantes | Variants Miner+ | Porciento de mejora: |
|---|---|---|---|
| Caminatas_700_Noisy_0.2pc.xes | 66319 ms | 62007 ms | 6.501907% |
| driveClass_700_Noisy_0.2pc.xes | 189451 ms | 145891 ms | 22.99275% |
| exampleLog_300_Noisy_0.2pc.xes | 74464 ms | 11547 ms | 84.49318% |
| groupedFollowsparallel5_700_Noisy_0.2pc.xes | 22578 ms | 11320 ms | 49.8627% |
| Promedio | | | 40.96263 % |

Los resultados demuestran que el algoritmo propuesto obtuvo una mejora de tiempo con respecto a la Minería de Variantes de aproximadamente un 40%. Los valores para los registros de eventos analizados oscilan entre un 7% y un 84%.

Las evaluaciones de calidad se obtuvieron mediante el marco de trabajo CoBeFra. Para las pruebas se utilizaron los 4 registros de eventos descritos anteriormente. Los resultados se comparan con los obtenidos por los modelos originales y por la Minería de Variantes ante los mismos registros de eventos, como se muestra en la Tabla 6.

***Tabla 6:*** *Resultados de las pruebas de calidad.*

| Registros de eventos | Precisión | | | Aptitud | | |
|---|---|---|---|---|---|---|
| | Original | VM | VM+ | Original | VM | VM+ |
| Caminatas_700_Noisy_0.2pc.xes | 0.99881 | 0.749991 | 0.412332 | 0.980971 | 0.987074 | 0.933028 |
| driveClass_700_Noisy_0.2pc.xes | 0.931019 | 0.344417 | 0.183115 | 0.913508 | 0.999284 | 0.999943 |
| exampleLog_300_Noisy_0.2pc.xes | 1 | 0.669284 | 0.407203 | 0.803969 | 0.947548 | 0.94496 |
| groupedFollowsparallel5_700_Noisy_0.2pc.xes | 0.997571 | 0.777346 | 0.311581 | 0.888113 | 0.471425 | 0.997551 |

En la tabla 6 es posible apreciar los resultados del *Variants Miner+* en la dimensión de aptitud. En algunos casos se obtuvieron valores superiores a los de los modelos originales. En la dimensión de precisión los resultados fueron menores. Sin embargo, la Minería de Variantes también presenta menores resultados en esta dimensión. Por tanto, al ser el *Variants Miner+* una derivación de la Minería de Variantes por transitividad se esperaban resultados menores. Ante estos resultados se puede apreciar que el algoritmo *Variants Miner+* prioriza la aptitud por encima de la precisión.

Malos resultados en la dimensión de precisión pueden ser interpretados como un caso de *underfitting.* O sea, que el modelo sobre generaliza o permite comportamientos muy diferentes de los que se muestran en el registro de eventos. Esto significa que además una evaluación en la dimensión de generalización probablemente hubiese dado buenos resultados. El motivo de estos resultados en la precisión es que no se considera comportamientos poco frecuentes. A continuación, se presenta un cálculo de los porcentajes de mejora de los resultados, en la Tabla 7.

***Tabla 7:*** *Análisis porcentual de los resultados obtenidos con respecto a la Minería de Variantes y a los modelos originales.*

| Registros de eventos | Precisión | | Aptitud | |
|---|---|---|---|---|
| | Original | VM | Original | VM |
| Caminatas_700_Noisy_0.2pc.xes | 41.28232 | 54.97826 | 95.11267 | 94.52464 |
| driveClass_700_Noisy_0.2pc.xes | 19.66826 | 53.16679 | 109.4619 | 100.0659 |
| exampleLog_300_Noisy_0.2pc.xes | 40.72028 | 60.84157 | 117.537 | 99.72689 |
| groupedFollowsparallel5_700_Noisy_0.2pc.xes | 31.23397 | 40.08267 | 112.3225 | 211.6034 |

Con respecto a la aptitud se puede apreciar que se obtuvo una depreciación de hasta un 5% con respecto a la Minería de Variantes y a los modelos originales. Las evaluaciones con respecto a ambos se mantuvieron muy similares. En el peor de los casos se obtuvo un resultado de un 95% para ambos. Hubo casos en los que el resultado obtenido para el *Variants Miner+* fue incluso un 211% del resultado obtenido por la Minería de Variantes.

Para la precisión los resultados oscilan entre un 40% y un 60% con respecto a la Minería de Variantes. Sin embargo, con respecto a los modelos originales se obtiene de un 20% a un 40%. Cabe destacar la Minería de Variantes también tiene bajos resultados en esta dimensión. Por tanto, es de esperar que el *Variants Miner+* también tenga problemas en la precisión.

## 3.4. Conclusiones parciales

En este capítulo se analizó la calidad de los modelos de procesos, la cual está enmarcada dentro de una de las tres áreas de la Minería de Procesos, el chequeo de conformidad. Este posee 4 principales dimensiones para medir la calidad de un modelo de proceso, dicha calidad refleja en qué medida el modelo representa el comportamiento contenido en el registro de eventos. Las cuatro dimensiones

son aptitud, precisión, generalización y simplicidad, para que un modelo de procesos tenga una buena evaluación de calidad debe obtener un buen resultado en cada una de ellas. Por tanto, los algoritmos de descubrimiento deben obtener modelos con un balance entre dichas dimensiones. De estas dimensiones se escogió la precisión y la aptitud para la validación del algoritmo propuesto.

Para evaluar el algoritmo se elaboró un caso de estudio compuesto de 4 registros de eventos con gran cantidad de sucesiones indirectas. Las dimensiones seleccionadas fueron evaluadas mediante métricas implementadas en el marco de trabajo CoBeFra. El algoritmo propuesto logró una disminución del tiempo de ejecución con respecto al algoritmo original de un 40%.

Sin embargo, la evaluación de la calidad para la dimensión de aptitud comparada con la Minería de Variantes oscila de un 95% a un 211%. Con respecto a los modelos originales la aptitud oscila de un 95% a un 117%. Para la dimensión de precisión se obtuvo un valor de un 40% a un 60% con respecto a la Minería de Variantes y de un 20% a un 40% con respecto a los modelos originales. Se puede apreciar que el algoritmo prioriza la aptitud de los modelos.

## CONCLUSIONES GENERALES

En el siguiente trabajo se analizaron los enfoques de solución utilizados por algunos algoritmos de descubrimiento de procesos ante registros de eventos con alto número de sucesiones indirectas entre las tareas. El desarrollo y validación de un algoritmo para disminuir el tiempo de obtención de variantes de proceso de negocio permitió llegar a las siguientes conclusiones:

- El enfoque predominante para tratar las sucesiones indirectas por parte de los algoritmos de descubrimiento consiste en descartar aquellas que sean menos frecuentes, ya sea en el análisis por cada patrón o para discernir entre patrones.
- La inserción de un umbral de tiempo permite disminuir la cantidad de comportamiento por procesar para la obtención de variantes de descomposición. Como la Minería de Variantes procesa el comportamiento en orden de su frecuencia se garantiza que el comportamiento procesado en ese momento es más frecuente que el comportamiento que queda por procesar
- Las pruebas realizadas evidencian que el algoritmo propuesto puede obtener variantes de proceso con una mejora de un 40% en el tiempo de ejecución independientemente del número de sucesiones indirectas presentes en el registro de eventos.
- Los experimentos realizados evidencian una depreciación de la precisión de un 20% a un 60%. Para la aptitud se obtuvieron evaluaciones de un 95% a un 211%. Los resultados demuestran que el algoritmo propuesto prioriza la aptitud sobre la precisión lo cual es coherente con el algoritmo original.

## RECOMENDACIONES

Como continuidad del presente trabajo el autor recomienda:

- Garantizar que durante el algoritmo de búsqueda de vecinos no se generen nodos que ya se habían descubierto previamente mediante el análisis de otro comportamiento.
- Disminuir la cantidad de patrones a considerar al descomponer un subproceso considerando que si se tiene una descomposición con una buena evaluación en un patrón es poco probable que a partir del comportamiento analizado se obtenga una mejor evaluación por otro patrón.
- Utilizar programación concurrente para descomponer en paralelo ramas diferentes del árbol de variantes.

## REFERENCIAS BIBLIOGRÁFICAS

ADRIANSYAH, A., MUNOZ-GAMA, J., CARMONA, J., VAN DONGEN, B.F. y VAN DER AALST, W.M., 2012. Alignment based precision checking. *International Conference on Business Process Management*. S.l.: Springer, pp. 137-149.

ADRIANSYAH, A., SIDOROVA, N. y VAN DONGEN, B.F., 2011. Cost-based fitness in conformance checking. *Application of Concurrency to System Design (ACSD), 2011 11th International Conference on* [en línea]. S.l.: IEEE, pp. 57–66. [Consulta: 30 junio 2016]. Disponible en: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5988918.

BEZERRA, F. y WAINER, J., 2012. A Dynamic Threshold Algorithm for Anomaly Detection in Logs of Process Aware Systems. *Journal of Information and Data Management*, vol. 3, no. 3, pp. 316.

BOSE, J.C., 2012. *Process Mining in the Large: Preprocessing, Discovery, and Diagnostics* [en línea]. PhD thesis. S.l.: Eindhoven University of Technology. Disponible en: http://www.win.tue.nl/jcbose/thesis/JC_Thesis.pdf.

BOSE, R.J.C., MANS, R.S. y VAN DER AALST, W.M.P., 2013. Wanna Improve Process Mining Results? [en línea], Disponible en: http://bpmcenter.org/wp-content/uploads/reports/2013/BPM-13-02.pdf.

BUIJS, J.C., VAN DONGEN, B.F. y VAN DER AALST, W.M.P., 2012. On the role of fitness, precision, generalization and simplicity in process discovery. En: 00008, *On the Move to Meaningful Internet Systems: OTM 2012* [en línea]. S.l.: Springer, pp. 305–322. Disponible en: http://link.springer.com/chapter/10.1007/978-3-642-33606-5_19.

CICCIO, C.D., MECELLA, M. y MENDLING, J., 2015. The Effect of Noise on Mined Declarative Constraints. En: P. CERAVOLO, R. ACCORSI y P. CUDRE-MAUROUX (eds.), *Data-Driven Process Discovery and Analysis* [en línea]. S.l.: Springer Berlin Heidelberg, Lecture Notes in Business Information Processing, 203, pp. 1–24. ISBN 978-3-662-46435-9. Disponible en: http://link.springer.com/chapter/10.1007/978-3-662-46436-6_1.

DE WEERDT, J., 2012. *Business process discovery: new techniques and applications.* [en línea]. S.l.: s.n. Disponible en: http://www.kuleuven.be/doctoraatsverdediging/cm/3H10/3H100298.htm.

DE WEERDT, J., BAESENS, B. y VANTHIENEN, J., 2013. A Comprehensive Benchmarking Framework (CoBeFra) for conformance analysis between procedural process models and event logs in ProM. *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2013), part of the IEEE Symposium Series in Computational Intelligence 2013* [en línea]. S.l.: s.n., Disponible en: http://eprints.qut.edu.au/57682/.

GÜNTHER, C. y VAN DER AALST, W.M.P., 2007. Fuzzy mining - Adaptive process simplification based on multi-perspective metrics. *Business Process Management* [en línea]. Brisbane: s.n., pp. 328-343. ISBN 03029743 (ISSN); 9783540751823 (ISBN). Disponible en: http://www.springerlink.com/index/n7610472h7680875.pdf.

LEEMANS, S.J., FAHLAND, D. y VAN DER AALST, W.M.P., 2013. Discovering Block-Structured Process Models From Event Logs Containing Infrequent Behaviour. *BPM Conference* [en línea]. S.l.: s.n., Disponible en: http://fluxicon.com/blog/wp-content/uploads/2013/09/Discovering-Block-Structured-Process-Models.pdf.

LEEMANS, S.J.J., FAHLAND, D. y VAN DER AALST, W.M.P., 2013. Discovering Block-Structured Process Models from Event Logs - A Constructive Approach. En: J.-M. COLOMy J. DESEL (eds.), *Application and Theory of Petri Nets and Concurrency* [en línea]. S.l.: Springer Berlin Heidelberg, Lecture Notes in Computer Science, 7927, pp. 311–329. ISBN 978-3-642-38696-1. Disponible en: http://link.springer.com/chapter/10.1007/978-3-642-38697-8_17.

LI, C., REICHERT, M. y WOMBACHER, A., 2011. Mining business process variants: Challenges, scenarios, algorithms. *Data and Knowledge Engineering*, vol. 70, no. 5, pp. 409–434. ISSN 0169023X (ISSN).

MITSYUK, A.A. y SHUGUROV, I.S., 2014. On Process Model Synthesis Based on Event Logs with Noise. *Modelirovanie i Analiz Informatsionnykh Sistem [Modeling and Analysis of Information Systems]*, vol. 21, no. 4, pp. 181–198.

PÉREZ-ALFONSO, D., 2015. *Método para el diagnóstico de procesos de negocio a partir de registros de eventos con ruido y ausencia de información.* La Habana, Cuba: Universidad de las Ciencias Informáticas.

PÉREZ-ALFONSO, D., PUPO-HERNÁNDEZ, E., YZQUIERDO-HERRERA, R. y LÓPEZ-JIMÉNEZ, R., 2015. Algoritmo para la identificación de variantes de procesos. ,

PUPO-HERNÁNDEZ, E. y LÓPEZ-JIMÉNEZ, R., 2014. *Algoritmo para la detección de patrones de control de flujo en los registros de eventos reales.* La Habana, Cuba: Universidad de las Ciencias Informáticas.

RECKER, J., REIJERS, H. y WOUW, S. van de, 2014. Process Model Comprehension: The Effects of Cognitive Abilities, Learning Style, and Strategy. *Communications of the Association for Information Systems* [en línea], vol. 34, no. 1. ISSN 1529-3181. Disponible en: http://aisel.aisnet.org/cais/vol34/iss1/9.

REDLICH, D., MOLKA, T., GILANI, W., BLAIR, G. y RASHID, A., 2014. Constructs Competition Miner: Process Control-Flow Discovery of BP-Domain Constructs. En: S. SADIQ, P. SOFFER y H. VOLZER (eds.), *Business Process Management* [en línea]. Cham: Springer International Publishing, pp. 134–150. ISBN 978-3-319-10171-2. Disponible en: http://link.springer.com/10.1007/978-3-319-10172-9_9.

ROSEN, K.H. y KRITHIVASAN, K., 1999. *Discrete mathematics and its applications*. S.l.: McGraw-Hill New York.

ROZINAT, A., MEDEIROS, A.K.A. de, GÜNTHER, C.W., WEIJTERS, A.J.M.M. y VAN DER AALST, W.M.P., 2007. Towards an Evaluation Framework for Process Mining Algorithms. *BPM Center Report* [en línea], Disponible en: BPMcenter.org.

RUSSELL, S.J., NORVIG, P., CANNY, J.F., MALIK, J.M. y EDWARDS, D.D., 1995. *Artificial intelligence: a modern approach* [en línea]. 2. S.l.: Prentice hall Englewood Cliffs. Prentice Hall Series in Artificial Intelligence. ISBN 0-13-790395-2. Disponible en: https://www.cis.uab.edu/courses/cs760/Spring-660-2007/7A-660-PLUS-SYLLABUS-DRAFT.pdf.

VAN DER AALST, W.M.P., 2011. *Process Mining. Discovery, Conformance and Enhancement of Business Processes*. S.l.: Springer, Heidelberg, Dordrecht, London et. al. ISBN 978-3-642-19344-6.

VAN DER AALST, W.M.P., 2011. *Process Mining: Discovery, Conformance and Enhancement of Business Processes* [en línea]. Berlin, Heidelberg: Springer Berlin Heidelberg. [Consulta: 30 junio 2016]. ISBN 978-3-642-19344-6. Disponible en: http://link.springer.com/10.1007/978-3-642-19345-3.

VAN DER AALST, W.M.P., 2013. Decomposing Petri nets for process mining: A generic approach. En: 00035, *Distributed and Parallel Databases*, vol. 31, no. 4, pp. 471–507. ISSN 0926-8782, 1573-7578. DOI 10.1007/s10619-013-7127-5.

VAN DER AALST, W.M.P., RUBIN, V., VERBEEK, H.M.W., VAN DONGEN, B.F., KINDLER, E. y GÜNTHER, C.W., 2010. Process mining: A two-step approach to balance between underfitting and overfitting. *Software and Systems Modeling*, vol. 9, no. 1, pp. 87–111. ISSN 16191366 (ISSN).

VAN DER AALST, W.M.P., WEIJTERS, A.J.M.M. y MARUSTER, L., 2004. Workflow Mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1128–1142. DOI 10.1109/TKDE.2004.47.

VÁZQUEZ-BARREIROS, B., MUCIENTES, M. y LAMA, M., 2015. ProDiGen: Mining complete, precise and minimal structure process models with a genetic algorithm. En: 00001, *Information Sciences*, vol. 294, pp. 315–333.

VERBEEK, H.M.W., BUIJS, J.C.A.M., VAN DONGEN, B.F. y VAN DER AALST, W.M.P., 2011. *XES, XESame, and ProM 6* [en línea]. Hammamet: s.n. CAiSE Forum 2010 on Information Systems Evolution. ISBN 18651348 (ISSN); 3642177212 (ISBN); 9783642177217 (ISBN). Disponible en: http://hinari-gw.who.int/whalecomwww.scopus.com/whalecom0/inward/record.url?eid=2-s2.0-79551529484&partnerID=40&md5=75bbb5c96bdc35cc9e1422ed2cd1cab4.

WEIJTERS, A., VAN DER AALST, W.M.P. y DE MEDEIROS, A.K.A., 2006. Process mining with the heuristics miner-algorithm. *Technische Universiteit Eindhoven, Tech. Rep. WP* [en línea], vol. 166. Disponible en: http://cms.ieis.tue.nl/Beta/Files/WorkingPapers/Beta_wp166.pdf.

WESKE, M. y HEIDELBERG, S.-V.B., 2007. *Business Process Management. Concepts, Languages, Architectures*. S.l.: s.n. ISBN 9783540735.

YANG, H., VAN DONGEN, B.F., TER HOFSTEDE, A.H.M., WYNN, M.T. y WANG, J., 2012. Estimating Completeness of Event Logs. En: LCCN: 0000 [en línea], Disponible en: http://bpmcenter.org/wp-content/uploads/reports/2012/BPM-12-04.pdf.

YZQUIERDO-HERRERA, R., 2012. *Modelo para la estimación de información ausente en las trazas usadas en la minería de proceso*. PhD Thesis. S.l.: Universidad de las Ciencias Informáticas.

YZQUIERDO-HERRERA, R., SILVERIO-CASTRO, R. y LAZO-CORTÉS, M., 2013. Tratamiento de la ausencia de información en la minería de procesos. *Revista Facultad de Ingeniería Universidad de Antioquia*, vol. 0, no. 69, pp. 67–78. ISSN 0120-6230.