Universidad de las Ciencias Informáticas

Título: Algoritmo para el diagnóstico de variantes de proceso en registros de eventos complejos

Autor: Alfredo Aguilera Miranda

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

Tutores: Ms.C. Damián Pérez Alfonso

Ing. Eudel Pupo Hernández

La Habana 2015

"No puedes seguir culpándote a ti mismo.
Cúlpate una vez y sigue adelante"

- Matt Groening

Dedicatoria

A Beatriz, mi hermanita querida:

... a quien amo con la vida, pero que me vuelve loco.

A mis padres:

... el sustento de mi vida.

Agradecimientos

Quisiera agradecer a todas las personas que de una forma u otra han ayudado a que hoy este aquí hablándoles. A mis padres, que se desviven por mí. A mi mamá, se acabaron los interrogatorios. A mi papá, que me enseño a programar, pero ya te doy tres vueltas y ni te enteras. A mi hermanita, que me regañó siempre que tuvo la oportunidad, ya me tocará a mí. A mi segunda casa y a mi primo Gabriel. A mi familia completa, que no se imaginan cuan numerosa es. A mi segunda familia, la Guerrilla de Malnombre, por enseñarme tantas cosas y tantos lugares. En especial a Eledys que sin su laptop ahora fuera por la introducción. Un agradecimiento especial a mis tutores. A Damián por tantas oportunidades y a Pupo por tanta paciencia. Gracias a los dos. Al tribunal, por la ayuda inmensa, y los comentarios oportunos. Gracias a todos. A mi hermanito del alma Alain. Donde quiera que estés nunca me olvidaré de ti. A Carlos, que me soportó y me tendrá que soportar por buen rato, gracias a ti. Y a la plebe: Andrés, Ariadni, Taylor, Omarito, Jomba, Mantilla, el Chepe. A todos mis amigos de todas partes. A todos los presentes.

¡Muchas Gracias!

RESUMEN

Las técnicas de descubrimiento de proceso ayudan a comprender el funcionamiento de las organizaciones a partir de sus procesos. Un algoritmo de descubrimiento toma como entrada un registro de eventos y descubre un modelo que representa el comportamiento del proceso. La técnica Minería de Variantes permite el diagnóstico de variantes de proceso, pero el costo computacional ante registros de eventos con presencia de ruido y ausencia de información aumenta significativamente. La solución propuesta consiste en complementar las deficiencias de Minería de variantes con las fortalezas del algoritmo de competición de la técnica Constructs Competition Miner. La validación mediante la evaluación de métricas de calidad sobre los modelos descubiertos por la solución propuesta muestra resultados favorables.

Palabras clave: ausencia de información, descubrimiento de proceso, registro de eventos, ruido.

ABSTRACT

Process Discovery techniques help to understand the actual processes deployed in an organization. A process discovery algorithm takes as input an event log and discovers a model that represents the actual process behavior. Variants Miner allows process variants diagnosis, but computational cost increases significantly when the event log contains noise and incompleteness. The solution proposed consists in complementing Variants Miner weaknesses with Constructs Competition Miner (CCM) competition algorithm strengths. The validation through evaluation of quality metrics on the models discovered by the proposed solution shows right results.

Keywords: event logs, incompleteness, noise, process discovery

ÍNDICE

DECLARACIÓN DE AUTORÍA

Declaro ser el único autor del presente trabajo que lleva como título: “Algoritmo para el diagnóstico de variantes de proceso en registros de eventos complejos” y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.
Para que así conste firmamos la presente a los ____ días del mes _____ del año _____.

Alfredo Aguilera Miranda

_____

Autor

MSc. Damián Pérez Alfonso

_____

Tutor

Ing. Eudel Pupo Hernández

_____

Tutor

Introducción

Las empresas y organizaciones cada vez dan mayor protagonismo a la gestión de sus procesos. Los Sistemas de Gestión de Procesos de Negocios (BPMS[1]) brindan funcionalidades para controlar la ejecución de los procesos, evaluar el desempeño de un sistema dado y apoyar el proceso de toma de decisiones.

Los sistemas de información registran 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.

El análisis de los datos compilados en los registros de eventos presenta una oportunidad para entender el funcionamiento real de una empresa a partir de los diferentes procesos que se llevan a cabo en ella y el orden en que se realizan. Con el fin de estudiar y aprovechar esta información surge la minería de proceso, disciplina que desarrolla técnicas y herramientas que permiten analizar los registros de eventos, extraer información a partir de ellos y presentar de una forma explícita el conocimiento que contienen (AALST., 2011).

La minería de procesos es una disciplina relativamente joven, la cual sirve como enlace entre la minería de datos y el modelado y análisis de procesos. La idea principal detrás de la minería de proceso es descubrir, monitorizar y mejorar procesos reales mediante la extracción de conocimiento de los registros de eventos que generan los sistemas de información.

Existen tres tipos de técnicas de minería de proceso: descubrimiento, chequeo de conformidad y mejora de procesos. Las técnicas de descubrimiento permiten modelar el comportamiento expresado en un registro de eventos. La aplicación de las técnicas de chequeo de conformidad permite comprobar si el modelo descubierto expresa el comportamiento real del proceso mediante la reproducción del registro de eventos. En el mejoramiento la idea principal es mejorar el modelo de proceso a partir de la información que brinda el registro de eventos para expresar con mayor veracidad el comportamiento real presente en el mismo. Las técnicas de minería de proceso han demostrado su utilidad en varias fases del ciclo de vida de la gestión de procesos de negocio. (AALST., 2011)

El ciclo de vida de la Gestión de Procesos de Negocio (BPM[2]) abarca las siete fases de un proceso de negocio, dentro de las cuales se encuentra el diagnóstico de proceso. La fase de diagnóstico de proceso comprende el análisis de rendimiento, la detección de anomalías y la identificación de patrones

[1] Del Inglés: Business Process Management Systems
[2] Del Inglés: Business Process Management

comunes (BOSE R.P.J.C., 2012). El diagnóstico ayuda a tener una visión general del proceso, de sus aspectos más significativos y de las técnicas que pueden ser más útiles en su posterior análisis (YZQUIERDO HERRERA, 2012).

En la fase de diagnóstico del proceso se han empleado técnicas de descubrimiento de proceso, las mismas consisten en determinar modelos de procesos representativos del comportamiento expresado en un registro de eventos. Con este fin han surgido diferentes algoritmos de descubrimiento que permiten obtener modelos de proceso en notaciones conocidas, entre las que se encuentran: Redes de Petri, Redes de Flujo de Trabajo y Redes Heurísticas.

El descubrimiento de proceso es una tarea compleja y con disímiles retos, los cuales influyen en la calidad de los modelos descubiertos (LY, 2012). 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). Asimismo, la presencia en el proceso de patrones de control de flujo como el lazo y la elección no libre, representan una dificultad para algunos de estos algoritmos (VAN DONGEN, 2009).

En registros de eventos que presentan características como las anteriormente descritas y a los que llamaremos complejos, se obtienen modelos de proceso que contienen gran cantidad de comportamiento. Este tipo de modelo de proceso se conoce como modelos poco estructurados y son estereotipados como modelos espaguetis (AALST. W. M., 2011). Un enfoque para el análisis de estos procesos poco estructurados es descubrir no un modelo, sino modelos alternativos del mismo proceso, denominados variantes de proceso (PÉREZ ALFONSO, 2014).

Los registros de eventos provenientes de entornos reales suelen presentar ruido, ausencia de información y un alto número de sucesiones indirectas, causando que los algoritmos de descubrimiento presenten dificultades en la identificación de los patrones de control de flujo. 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 patrones de control de flujo en un proceso posibilita mejorar la comprensión del funcionamiento general del mismo. Las técnicas desarrolladas en el área de diagnóstico presentan problemas con la identificación de los patrones de control de flujo más comunes, las desviaciones en el registro de eventos y la determinación de diferentes niveles de jerarquización entre las actividades.

A partir de la situación problemática expuesta anteriormente se plantea como **problema a resolver**: La presencia en los registros de eventos de ruido y ausencia de información dificultan el descubrimiento de variantes de proceso.

Se delimita como **objeto de estudio** la minería de proceso.

Como **objetivo general** de este trabajo se plantea: Desarrollar un algoritmo de descubrimiento de procesos para posibilitar el diagnóstico de variantes de proceso en registros de eventos con ruido y ausencia de información.

Como **campo de acción** se delimita el descubrimiento de proceso.

El objetivo general se puede desglosar en los siguientes **objetivos específicos**:

- Analizar los enfoques de solución utilizados por los algoritmos de descubrimiento de proceso ante registros de eventos con ruido y ausencia de información.
- Implementar un algoritmo que posibilite el diagnóstico de variantes de proceso en registros de eventos con ruido y ausencia de información.
- Validar el algoritmo implementado a partir del diagnóstico de variantes de proceso en registros de eventos con ruido y ausencia de información.

Como **posibles resultados** a obtener mediante la realización de este trabajo se proponen:

- Un algoritmo para el diagnóstico de variantes de proceso en registros de eventos con ruido y ausencia de información.
- La implementación del algoritmo diseñado dentro del marco de trabajo de la minería de procesos ProM.

Durante el desarrollo del presente trabajo se definen las siguientes tareas específicas de la investigación:

- Análisis de los principales conceptos y trabajos relacionados con el descubrimiento de 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 diagnóstico de variantes de proceso en registros de eventos con ruido y ausencia de información.
- Diseño de un algoritmo para el diagnóstico de variantes de proceso en registros de eventos con ruido y ausencia de información.
- Implementación del algoritmo diseñado permitiendo su integración a un marco de trabajo existente para la minería de procesos.

- Evaluación del algoritmo y la herramienta desarrollada.

El presente documento está estructurado en 4 capítulos cuyos contenidos se sintetizan a continuación.

- Capítulo 1: Se exponen los principales conceptos relacionados con la minería de proceso y el descubrimiento de proceso. Se realiza un estudio crítico de las técnicas de diagnóstico de proceso existentes, sus fortalezas y debilidades.

- Capítulo 2: Se presenta la propuesta de solución. Se describe el algoritmo diseñado analizando cada uno de los patrones de control de flujo descubiertos.

- Capítulo 3: Se presenta la implementación del algoritmo como un componente del marco de trabajo ProM. Se muestran los artefactos generados para la mejor comprensión y documentación del proceso de desarrollo.

- Capítulo 4: En este capítulo se presenta el experimento diseñado para la validación de la solución. Se exponen y analizan los resultados de la validación.

Capítulo 1

1.1 Minería de proceso.

La Minería de proceso es una disciplina científica cuyo fin es descubrir, monitorear y mejorar los procesos reales, a través de la extracción de conocimiento de los registros de eventos, disponibles en los sistemas de información. Esta se enmarca dentro de la Inteligencia de Negocios (BI[3]) formando una conexión entre la Minería de Datos y el Modelado y Análisis de Procesos (van der Aalst, 2011).

El proceso de negocio es un concepto básico tanto en la minería de proceso como en la BI. Se han ofrecido diversas definiciones sobre proceso de negocio (Hammer and Champy, 1993, Johansson et al., 1993, Rummler and Brache, 2012, Davenport, 2013). Para la presente investigación se asume la definición de Mathias Weske, por incluir los elementos de interés para la BI: "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 ejecutado por una simple organización, pero con él pueden interactuar procesos de negocios de otras organizaciones" (Weske and Heidelberg, 2007).

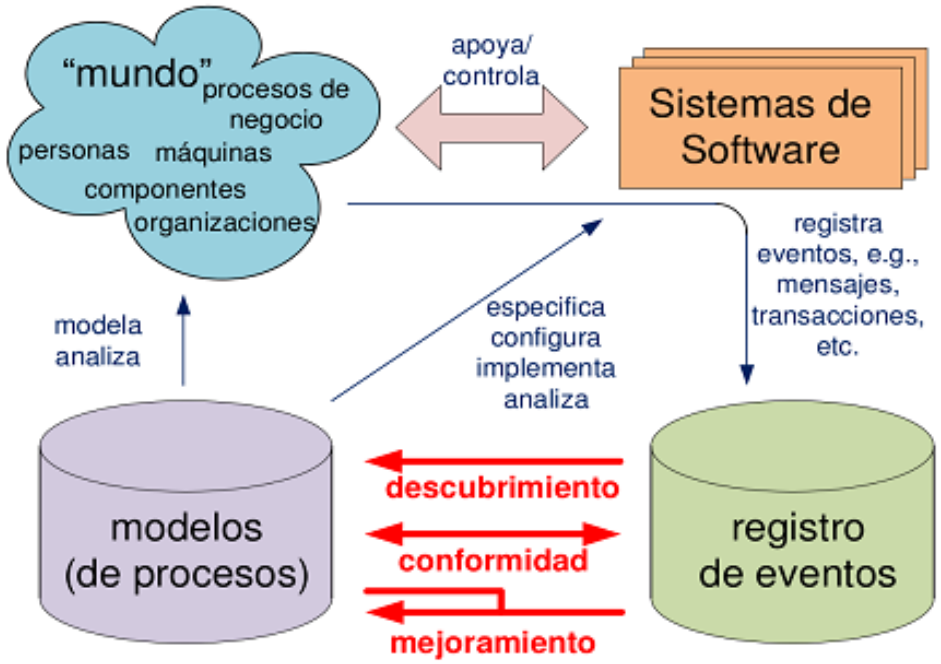


Figura. 1. Técnicas de minería de proceso en el ámbito de BPMN (AALST., 2011).

[3] Del Inglés: Business Intelligence

Las técnicas de minería de proceso permiten extraer información útil y no trivial de los registros de eventos almacenados por los sistemas de información. Mediante el empleo de estas técnicas se pueden obtener recomendaciones y predicciones, teniendo en cuenta el análisis de los datos actuales e históricos. Estas pueden clasificarse en tres grupos como se muestra en la Figura 1. El primer grupo engloba las destinadas al descubrimiento de modelo de procesos, el segundo representa el chequeo de conformidad (AALST et al. 2011) y el tercero abarca la extensión de un modelo existente.

1.2 Descubrimiento de proceso.

El descubrimiento de proceso consiste en determinar, a partir de un registro de eventos, un modelo de proceso que explique el comportamiento presente en el mismo (AALST., 2011). La principal motivación del descubrimiento de proceso es la utilidad de los modelos obtenidos a partir de la ejecución real de un proceso. Los modelos descubiertos pueden ser representados utilizando múltiples notaciones, dentro de las que se encuentran Redes de Petri y Redes de Flujo de Trabajo (AALST., 2011).

El descubrimiento de proceso permite detectar los patrones comunes presentes en un registro de eventos, así como la existencia de desviaciones en el proceso (BOSE J. C. R. P., van der AALST, 2012). El descubrimiento presenta una oportunidad para obtener una visión general de las principales características del proceso evidenciadas en el registro de eventos.

Mediante el empleo de técnicas de descubrimiento de proceso en el registro de eventos es posible conocer los comportamientos presentes de mayor frecuencia, lo que posibilita dirigir las técnicas de mejora hacia los elementos más críticos del proceso. Al mismo tiempo, la detección de ejecuciones anómalas existentes en el registro de eventos brinda información acerca de posibles violaciones del modelo. La presencia de desviaciones en el modelo constituye un indicador de posibles fraudes o violaciones en las políticas y suele resultar de utilidad a las organizaciones en la realización de auditorías. El descubrimiento de proceso permite identificar el contexto en el que se ejecuta una actividad en el registro de eventos y reconocer la existencia de dependencias entre las actividades que conforman el proceso (PUPO HERNÁNDEZ, LÓPEZ JIMÉNEZ, 2014).

Durante la fase de descubrimiento resulta importante determinar la presencia en el registro de eventos de patrones de control de flujo. En este sentido, uno de sus principales objetivos es la determinación de patrones como el paralelismo y la selección, relacionados con la ejecución simultánea de actividades, debido al incremento de complejidad que significa su posterior análisis (BOSE J. C. R. P., van der AALST, 2012).

El Descubrimiento de Proceso es una de las tareas de minería de proceso con más retos. Los registros de eventos presentan características como ruido, ausencia de información y/o presencia de patrones complejos. Los algoritmos de descubrimiento de procesos no manejan estas características en la misma medida, provocando, en algunos casos, la obtención de modelos de baja calidad.

1.3 Técnicas de descubrimiento de proceso

El problema más complejo de la minería de proceso es el de descubrir un modelo de proceso a partir de las trazas de un registro. Muchas técnicas de descubrimiento de proceso se han propuesto. Desafortunadamente, las técnicas existentes pueden producir modelos que son incapaces de reproducir el registro de eventos que los originaron. Pueden además generar modelos erróneos e incluso tener tiempos de ejecución excesivamente largos.

Qué modelo es "el mejor", se define teniendo en cuenta criterios de calidad establecidos. Un criterio importante es la **correctitud**. Un modelo es **correcto** si para todas las ejecuciones existe un estado final satisfactorio. Otro criterio de calidad es la aptitud. Un modelo tiene perfecta aptitud respecto a un registro de eventos si es capaz de reproducir todas las trazas en el mismo. El criterio de **precisión** expresa si un modelo no permite más comportamiento que el que contiene el registro de eventos. La **generalización** es la característica de los modelos que determina cuanto más comportamiento permitirá el modelo, aparte del que contiene el registro de eventos.

Algunos algoritmos garantizan la **correctitud** de un modelo y otros garantizan la aptitud. No obstante, no existe un algoritmo de descubrimiento que garantice la obtención de un modelo **apto** y **correcto** en un tiempo finito de ejecución para todos los registros de eventos que se le proporcionen.

Definiciones necesarias

Registro de eventos. Se asume **Σ** como el conjunto de todas las actividades de un proceso. Un evento **e** es la ocurrencia de una actividad: **e ∈ Σ**. Una *traza* **t** es una secuencia, puede ser vacía, de actividades: **t ∈ Σ∗.** Se denota la traza vacía con **ε**. Un registro de eventos **L** es una colección no vacía de trazas: **L ⊆ Σ∗**. Por ejemplo, $\{\langle a, b, c\rangle, \langle a, c, b\rangle\}$ denota un registro compuesto por dos trazas **abc** y **acb**. El *tamaño* de un registro es el número de eventos en él: **|| L||** = $\sum_{t\in L} |t|$ (PÉREZ ALFONSO, 2014).

Patrón de control de flujo: Un proceso se puede descomponer en múltiples subprocesos utilizando los siguientes patrones de control de flujo:

- Secuencia: dos subprocesos se ejecutan secuencialmente si uno ocurre inmediatamente después del otro.

- Selección exclusiva (XOR): dos subprocesos forman parte de una selección exclusiva si, en un punto de decisión, se puede ejecutar solamente uno de los dos.
- 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.

Subproceso: 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).

Variantes de proceso: Las variantes de un modelo de procesos o variantes de proceso, son modelos de un proceso que describen el mismo proceso de negocio, y poseen algunas 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, 2014).

Descomposición en subprocesos: Se denota con SP al conjunto de los subprocesos del proceso P, y por W = {ws, wx, wo, wp, wl} se denota al conjunto de los patrones de control de flujo. Un subproceso si $\in$ SP se puede descomponer a través de diferentes wj|$\forall$wj $\in$ W en varios sk|$\forall$k $\neq$ i : sk $\in$ SP, hasta el nivel de actividad (PÉREZ ALFONSO, 2014).

1.3.1 Constructs Competition Miner (CCM)

El enfoque de este algoritmo de descubrimiento consiste en asumir que un modelo de proceso de negocio está compuesto por un conjunto de bloques. Estos bloques son, entre otros, la secuencia, la elección, el ciclo y el paralelismo. El modelo descubierto se logra a partir de asumir que el modelo es una estructura anidada de estos bloques y dejar que los mismos compitan entre sí por cada nivel por la solución más adecuada desde lo más simple hasta lo más complejo. El aspecto competitivo del CCM lo hace especialmente adecuado para registros de eventos con comportamientos excepcionales y conflictivos entre sí.

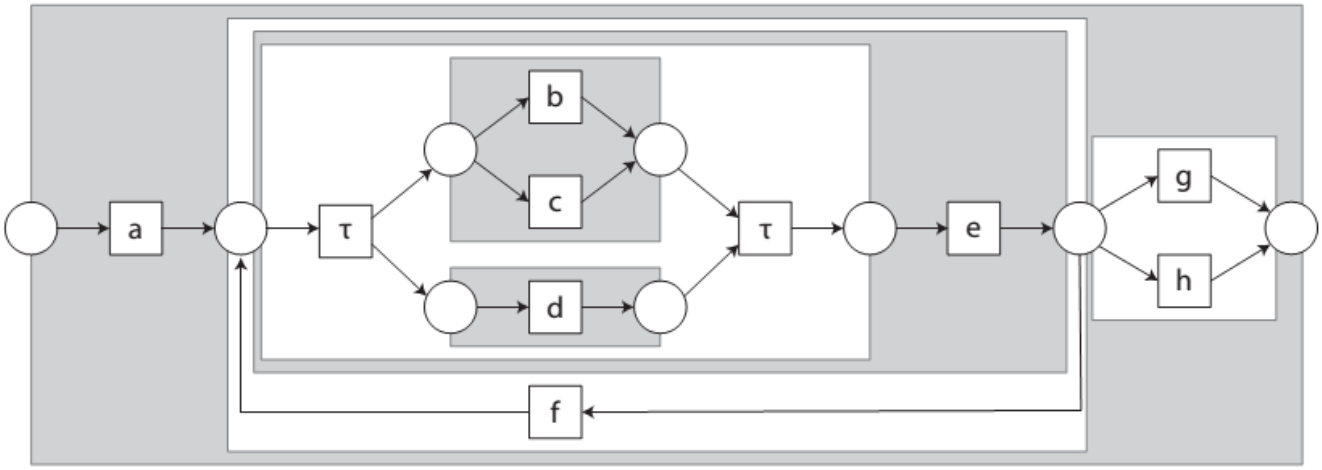


Figura. 2. Estructura anidada de bloques (van der Aalst, y otros, 2013).

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.

El CCM tiene el enfoque "divide y vencerás" recursivo. Primero se crea una matriz de causalidad que abarca a todas las actividades del registro de eventos. Basado en esta matriz se calcula el ajuste entre cualquier par de actividades hacia cada uno de los patrones de control de flujo disponibles (secuencia, elección, lazo, etc.). Este cálculo de ajuste está basado en la evaluación de restricciones para las relaciones globales entre las actividades que se observan en la matriz de causalidad. El objetivo del posterior algoritmo de competición es determinar la mejor combinación entre (1) el patrón de control de flujo y (2) el mejor par de conjuntos de actividades A1 y A2, que se ajuste al mismo. Si se decide qué patrón utilizar, el mismo es creado. Por ejemplo un bloque AND si el paralelismo resulta ganador en la evaluación de ajuste. A partir de llamadas recursivas se analizan los conjuntos A1 y A2 de la misma manera anteriormente descrita; se crea la matriz, se calcula el ajuste, etc. Para cada bloque se requiere que el conjunto se pueda dividir en dos conjuntos no vacíos. La recursividad continua hasta que no se pueda dividir más, en cuyo caso el método devuelve una señal de éxito luego de crear un bloque con una sola actividad. El proceso es construido completamente si la llamada inicial devuelve éxito.

El algoritmo soporta la identificación de los siguientes patrones de control de flujo:

- Secuencia
- Elección
- Paralelismo
- Lazo
- Secuencia en Lazo
- Elección en Lazo
- Paralelismo en Lazo

Para cada patrón se formulan un conjunto de restricciones que determinan el grado de ajuste del patrón y los subconjuntos de actividades. Existen tres tipos de restricciones diferenciadas en el grado de confianza que representa su cumplimiento y la penalización que supone su incumplimiento.

- *Estricto*: Las restricciones que entran en esta categoría pueden ser vistas como condiciones de tipo "si, y solo si", siendo requeridas para la identificación del patrón de flujo. El incumplimiento de este tipo de restricciones tiene una elevada penalización en el cálculo del ajuste del patrón correspondiente.
- *Comportamiento total*. Una restricción de tipo Comportamiento total se satisface si todo el comportamiento expresado por el patrón de control de flujo esta expresado en el registro de eventos. El incumplimiento de este tipo de restricciones tiene una penalización media en el cálculo del ajuste del patrón correspondiente.
- *Indicación*. Las restricciones de este tipo representan comportamientos típicos de un patrón de control de flujo que no tiene necesariamente que estar representado en el registro de eventos. Estas restricciones representan aproximaciones del comportamiento típico de un patrón, y permiten distinguir entre patrones similares. El incumplimiento de este tipo de restricciones tiene una penalización baja en el cálculo del ajuste del patrón correspondiente.

1.3.1.1 Algoritmo de competición

El objetivo del algoritmo de competición es encontrar la mejor combinación entre el patrón de control de flujo y el par de conjuntos de actividades **A1** y **A2** del total de actividades del proceso (**A**), de modo

que **A1 ∪ A2 = A** y **A1 ∩ A2 = {}**, que mejor se acomoda a las restricciones evaluadas sobre las relaciones entre todos los pares de actividades **x-y**.

Una solución sencilla sería crear y comparar todas las posibles divisiones de A. En lo que respecta al tiempo de ejecución del CCM, esto no es adecuado, debido a que se debería comparar $\mathbf{2^{|A|} - 1}$ posibles divisiones. En cambio se toma partido del hecho de que se analizan las relaciones globales entre las actividades. Esto significa que es irrelevante para el análisis de un par de actividades si pertenecen a un mismo conjunto o no.

El algoritmo1 muestra cómo trabaja el algoritmo de competición del CCM si se considera únicamente al patrón Selección como parte de la competencia. La cola con prioridad **openCases** se ordena primeramente por la penalidad del ajuste a las restricciones y luego por cuan parejos están los subconjuntos **A1** y **A2**, dado que se desea dividir lo más parejo posible para reducir de manera más rápida el número de actividades en las subsiguientes divisiones.

Partiendo del registro de eventos de ejemplo con **A = {a, b, c, d, e, f, g, h}** el algoritmo trabaja como sigue:

Se crea una tupla **{A1, A2, AR, p}** y se inserta en la cola. Con **A1** y **A2** vacíos al inicio, **AR** es el conjunto de actividades que no han sido asignadas todavía a **A1** o **A2** y **p** es la penalidad que tiene la división en cuanto a la evaluación de las restricciones para **Selección** en este caso.

A continuación se entra en el ciclo en el cual la primera tupla de la cola es extraída (en este caso la única que existe) y posteriormente procesada. Se extrae una actividad **x** de **AR** y se crean dos tuplas con **x** en **A1** y en **A2**. Dependiendo de en cual conjunto se encuentre **x** se calcula el ajuste de **x** hacia cada una de las actividades de los subconjuntos creados y se calcula un valor de penalidad **p**. Las dos tuplas creadas se insertan en la cola. El mismo proceso continua hasta que no queden más actividades por asignar (**AR = {}**). En la figura 3 se muestran las combinaciones creadas para el ejemplo, las combinaciones coloreadas de gris claro se encuentran en la cola todavía al momento de concluir el algoritmo. Las combinaciones de gris ya han sido procesadas y el número que tienen al lado representa el orden en que fueron procesadas. La combinación coloreada de negro es la ganadora del algoritmo de competición.

Más patrones de control de flujo pueden entrar en la competición si se hacen pequeñas modificaciones al algoritmo. La tupla en la cola con prioridad además debe contener el tipo de patrón, y se debe crear una tupla por cada patrón antes de entrar en el ciclo.

Algoritmo1: Algoritmo de competición considerando solo Selección (Ch)

Entrada: A, Ch

Salida: A1, A2

begin

```
ColaConPrioridad openCases ← {};
openCases.add(({}, {}, A, .0));
while true do
      (A1, A2, AR, p) ← openCases.poll();
      if AR = {} then return (A1, A2);
      x ← AR.poll();
      if |AR| > 0 V |A2| > 0 then
            Aaux ← A1 U {x};
            paux ← 0;
            foreach y ∈ A2 do paux ← paux + Ch(x, y);
            if paux > 0 then paux ← paux/|A2| + p;
            else paux ← p;
            openCases.add((Aaux, A2, AR, paux));
      end
      if |AR| > 0 V |A1| > 0 then
            Aaux ← A2 U {x};
            paux ← 0;
            foreach y ∈ A1 do paux ← paux + Ch(y, x);
            if paux > 0 then paux ← paux / |A1| + p;
            else paux ← p;
            openCases.add((A1, Aaux, AR, paux));
      end
end
```

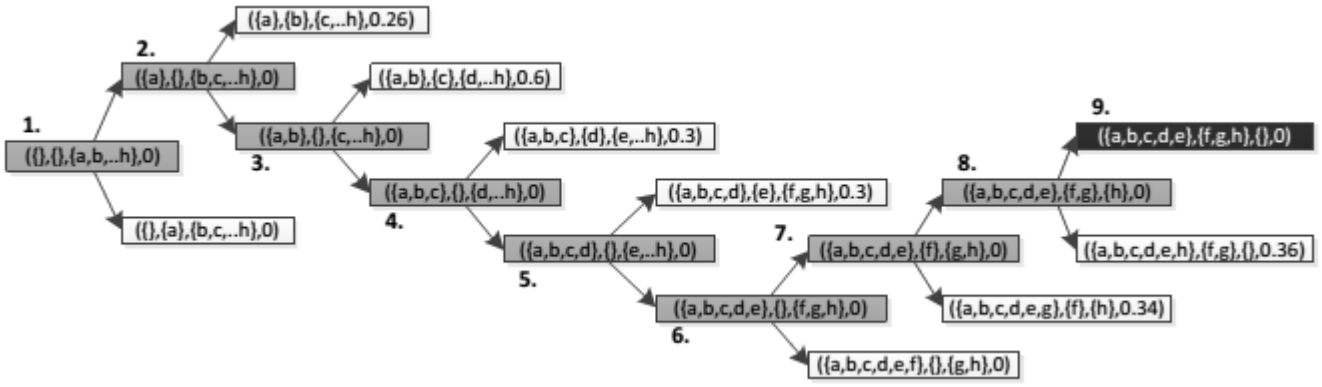


Figura. 3. Distintas tuplas hasta encontrar la mejor combinación.

Ventajas

- Enfoque “divide y vencerás” recursivo.
- Permite descubrir modelos estructurados.
- Su algoritmo de competición reduce el impacto del ruido en los modelos descubiertos.
- Extrae los comportamientos globales de las actividades.

Desventajas

- El algoritmo de competición obtiene solo una descomposición para el subproceso analizado.
- No existe una herramienta que implemente el algoritmo de competición.

1.3.2 Inductive Miner (IM)

La técnica IM permite descubrir modelos en forma de árboles de proceso. 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 como se combinan sus hijos.

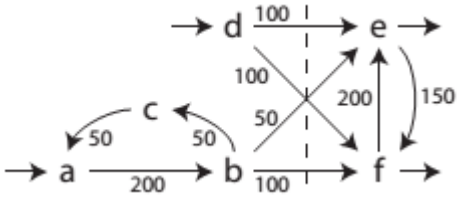
Se consideran los siguientes operadores: $\rightarrow$, $\times$, $\wedge$, $\circlearrowleft$. El operador $\rightarrow$ denota la secuencia entre sus hijos, $\times$ representa la relación de exclusión mutua, $\wedge$ se utiliza para representar el paralelismo y $\circlearrowleft$ es el lazo.

IM trabaja de manera recursiva, seleccionando el operador raíz que mejor se ajusta a un registro, 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 (van der Aalst, y otros, 2013).

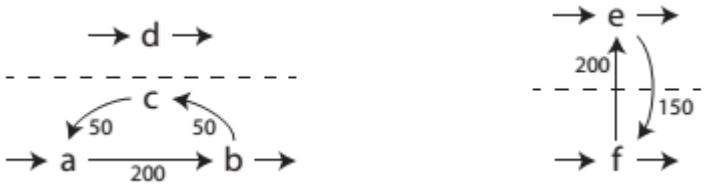
La siguiente figura muestra un grafo de sucesiones directas. IM busca una división en dos conjuntos disjuntos haciendo un corte en el grafo.

L: [<a, b, c, a, b, e, f> 50, <a, b, f, e> 100, <d, e, f> 100, <d, f, e> 100]



La línea a rayas representa un corte →, ya que todas las aristas que lo cruzan inician en la derecha y terminan a la izquierda del mismo. El registro queda dividido en los subconjuntos {a, b, c, d}, {e, f}, creándose los siguientes sub-registros: L 1 = [<a, b, c, a, b> 50, <a, b> 100, <d> 200] y L 2 = [<e, f> 150, <f, e> 200].

Cada uno de los operadores tiene un corte característico para el grafo de sucesiones directas. Si dicho corte se puede hacer IM selecciona el operador correspondiente, de lo contrario se obtiene un modelo en flor, que permite la ejecución de todas las secuencias de actividades.



IM aplica filtros al grafo de sucesiones directas para tratar el **ruido**. Se determina un factor k y se eliminan todas las aristas cuya frecuencia sea menor que k multiplicado por la frecuencia de la arista con mayor peso. Mediante este enfoque es posible eliminar del grafo las relaciones que son poco frecuentes y que podrían ser consideradas como ruido. Cuando esto no es posible dado que todas las relaciones tengan una frecuencia similar, es posible aplicar otros enfoques para eliminar el comportamiento poco frecuente.

IM utiliza las relaciones de causalidad para tratar con las actividades poco frecuentes. En un grafo de causalidad una arista entre **a** y **b** aparecerá si en alguna traza **a** es eventualmente seguido por **b**. De esta manera los pesos de las aristas son amplificados excepto el de las actividades poco frecuentes con lo que es posible aplicar los filtros y eliminar estas relaciones.

Algoritmo

Dado un conjunto de operadores ⊕ y un registro de eventos **L**, IM busca dividir **L** en subconjuntos más pequeños **L1** ... **Ln** tales que combinados con algún operador produzcan a **L**. La recursividad prosigue en los subconjuntos encontrados hasta que **L** no se pueda dividir más. Se hace un pequeño cambio al algoritmo dado que la idea de que **L** sea estrictamente dividido en subconjuntos que siempre sean más pequeños traería consigo que algunos modelos no sean descubiertos con precisión. De hecho se permite que **L** se pueda dividir en subconjuntos con el mismo tamaño, no obstante estas divisiones que no reducen el tamaño de **L** deben ocurrir una cantidad finita de veces. Para esto se introduce un parámetro **φ** el cual deberá ser decrementado cada vez que una división de este tipo ocurre. Este parámetro **φ** es en esencia una cota para la cantidad de ramas invisibles que se puedan derivar de analizar el registro de eventos.

```
function Bselect(L, φ)
    if L = {ε} then
```

```
        base ← {τ}
    else if ∃a ∈ Σ : L = {⟨a⟩} then
        base ← {a}
    else
        base ← ∅
    end if
    P ← select(L)
    if |P| = 0 then
        if base = ∅ then
            return {(τ, a1, ..., am) where {a1 • • • am} = Σ(L)}
        else
            return base
        end if
    end if
    return {⨁(M1, . . . , Mn)|(⨁, ((L1, φ1), ..., (Ln, φn))) ∈
P∧∀i : Mi ∈ B(Li, φi)}∪base
end function
```

1.3.3 Minería de Variantes

Minería de variantes es una técnica de descubrimiento de proceso. La misma tiene como entrada un registro de eventos y obtiene un árbol de variantes de proceso y un perfil de diagnóstico. Esta técnica está conformada por cuatro etapas que se ejecutan secuencialmente:

1. Pre-procesamiento del registro de eventos.

2. Extracción de comportamiento.

3. Búsqueda de variantes.

4. Generación del perfil de diagnóstico.

El objetivo perseguido por la Minería de Variantes es construir variantes de proceso a partir de diferentes descomposiciones en subprocesos considerando patrones de control de flujo. Las variantes de descomposición obtenidas se muestran en un Árbol de Variantes, asociado al cual se obtiene un Perfil de Diagnóstico (PÉREZ ALFONSO, 2014).

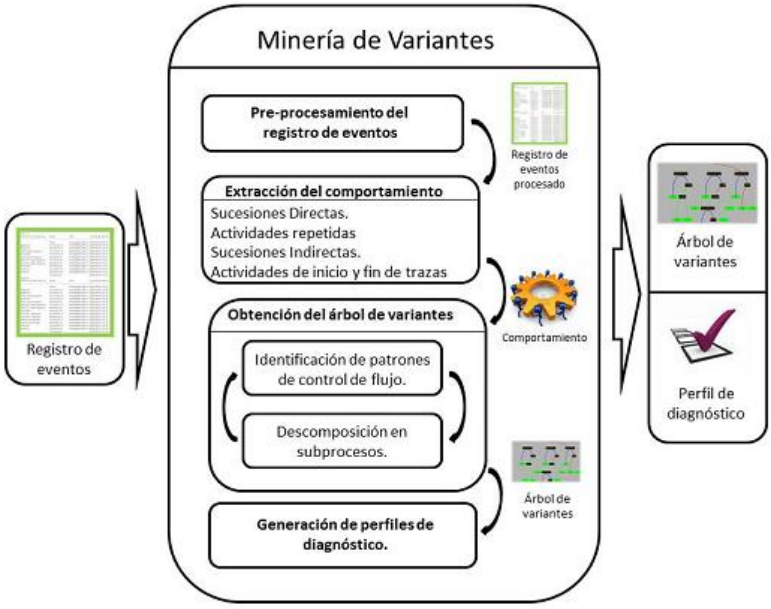


Figura 4. Minería de variantes (PÉREZ ALFONSO, 2014).

Esta técnica sigue un enfoque diferente al de otras técnicas de descubrimiento, al proponer varias descomposiciones alternativas para el mismo subproceso, utilizando diferentes patrones 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, descartando o no, determinados comportamientos presentes en el registro de eventos. También se pueden construir, considerando o no, determinados comportamientos ausentes. Las diferentes alternativas de descomposición que pueden existir en cada subproceso conforman variantes del proceso.

Variantes de proceso: Las variantes de un modelo de procesos o variantes de proceso, son modelos de un proceso que describen el mismo proceso de negocio, y poseen algunas 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.

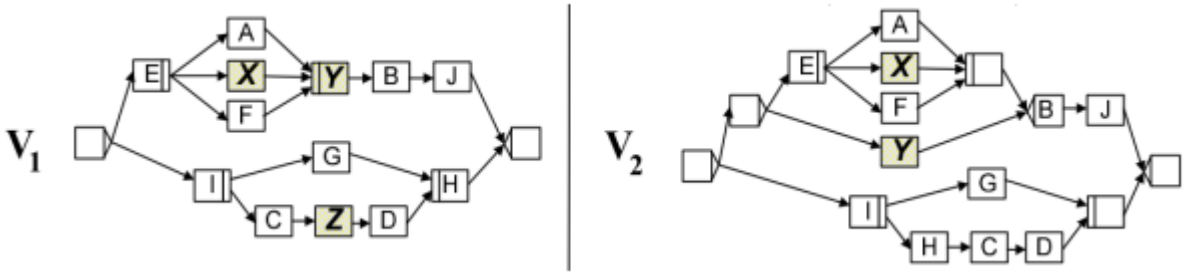


Figura 5. Variantes de proceso (PÉREZ ALFONSO, 2014).

En la Figura 5, pueden apreciarse dos variantes (V1 y V2) de un proceso de negocio. En 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.

Los elementos antes planteados permiten que la técnica ofrezca un modelo jerárquico en forma de árbol, que representa diferentes variantes del proceso cuya ejecución está contenida en el registro de eventos. Para esto se propone un modelo de proceso denominado Árbol de Variantes. Este árbol está compuesto por dos tipos de nodos, los nodos subproceso y los nodos 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.

En la Figura 6 se muestran las variantes de descomposición para un mismo subproceso, que forman parte de un Árbol de variantes.

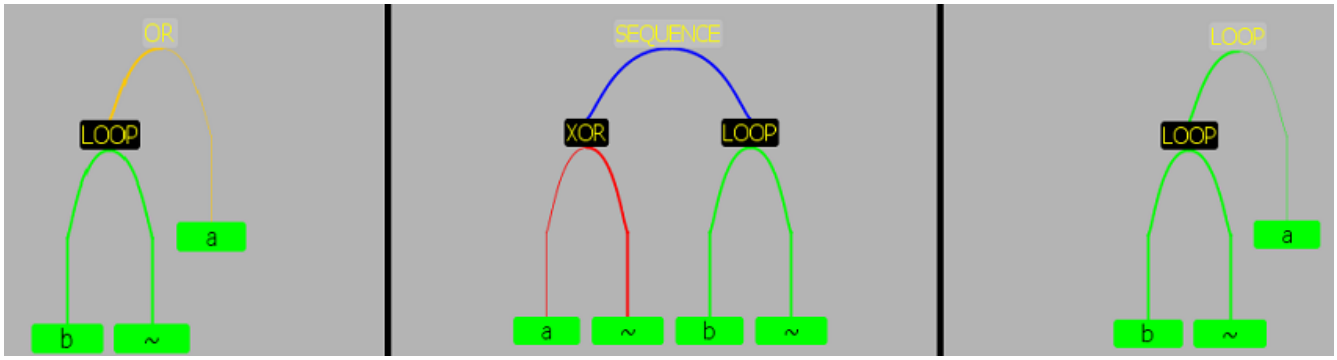


Figura 6. Variantes contenidas en un Árbol de variantes (PÉREZ ALFONSO, 2014).

La primera variante corresponde al patrón Selección no Exclusiva. La segunda variante representa una descomposición por el patrón de control de flujo Secuencia. Por último, se muestra la descomposición del subproceso utilizando el patrón Lazo. Los nodos hojas representan las actividades que conforman el proceso. Los nodos patrón, muestran el nombre del patrón de control de flujo utilizado para descomponer el subproceso en la variante correspondiente. Cada nivel en el árbol, de la raíz a las hojas, representa un nivel de abstracción inferior. Asociado al Árbol de Variantes se propone un Perfil de Diagnóstico, el cual agrupa la información de diagnóstico asociada a cada nodo de tipo patrón del árbol.

En el Perfil de Diagnóstico se incluyen los comportamientos descartados, ya que estos pueden ser considerados como ruido, anomalías o desviaciones del proceso. A su vez, los comportamientos

asumidos durante la descomposición, reflejan situaciones de ausencia de información, las cuales pueden estar asociadas también a anomalías durante la ejecución del proceso. La presentación de ambos tipos de comportamiento como resultado de la técnica, responde directamente a los intereses del diagnóstico. Se pretende, de esta manera, contribuir a la comprensión del proceso en la dimensión pragmática.

1.3.3.1 Pre-procesamiento del registro de eventos

Esta etapa tiene como objetivo preparar el registro de eventos para el diagnóstico de variantes. El algoritmo que se ejecuta en esta etapa realiza la codificación de los eventos presentes en el registro de eventos. La codificación consiste en la asignación de una letra a cada evento contenido en el registro. Esto tiene como objetivo disminuir la carga cognitiva durante la comprensión del proceso, eliminando la información semántica de las etiquetas de los eventos. Mediante la codificación se pretende mejorar la comprensión sintáctica del modelo, antes de pasar a la comprensión semántica del mismo. Adicionalmente, la conversión de las trazas en secuencias codificadas facilita el procesamiento subsecuente.

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.

1.3.3.2 Extracción de comportamiento

El propósito de esta 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 siguientes comportamientos

- *Sucesiones directas (DS),* se denomina sucesión directa entre la actividad **a** y la actividad **b**, a la existencia en alguna traza de la secuencia **ab**.
- *Sucesiones indirectas (IS),* 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.*
- *Eventos repetidos en las trazas.*

Por cada comportamiento extraído se obtiene la frecuencia de su aparición en las trazas codificadas. 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, cuya baja frecuencia de ejecución sugiere que deban ser considerados como ruido (PÉREZ ALFONSO, 2014).

1.3.3.3 Obtención de variantes de descomposición

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 de la combinación de los comportamientos extraídos en la fase anterior y la asunción de ciertos comportamientos ausentes del registro de eventos. En esta fase se ejecutan dos algoritmos iterativamente, uno para la descomposición en subprocesos y otro para la identificación de patrones de control de flujo.

Las posibles variantes de descomposición de un subproceso se determinan dentro de un ámbito definido como parámetros de entrada. El ámbito se especifica en términos de umbrales para ruido y completitud. A partir de un registro de eventos y un ámbito definido se genera el árbol de variantes.

Descomposición en subprocesos

El algoritmo de descomposición en subproceso utiliza un registro de eventos del proceso P y los umbrales definidos para construir un árbol de variantes VP. Siendo SP el conjunto de todos los subprocesos del proceso P, para cada nuevo subproceso si $\in$ SP, se busca una variante de descomposición según cada uno de los patrones de control de flujo.

Algoritmo Descomposición en subprocesos

Entrada: registro de eventos (LP), umbrales de ruido y completitud (TP)

Salida: árbol variantes (VP), perfil de diagnóstico

```
Procedimiento DECOMPOSE(LP, TP)

    W ← {ws, wx, wo, wp, wl} //Patrones de control de flujo para la
            descomposición

    ns1 ← CREATESUB-PROCESS(LP)  //Crea el nodo raíz.

    VP ← ns1

    SP ← {ns1}  //Crea una lista para los subprocesos nuevos.

    while nsi ← extraer el primer nodo subproceso existente en SP do

    if si no es una actividad then

        βli ← EXTRACTBEHAVIOR(nsi) //Extrae el comportamiento contenido
                en li

        for all wj en W do

            V ← FINDVARIANTS(wj, βli , TP)

            if |V| > 0 then //si nsi puede ser descompuesto por wj bajo
                    TP

                for all nodo subproceso k en V do

                    adicionar el nodo subproceso k como hijo del nodo
                            patrón niwj

                    adicionar el nodo subproceso k to SP

                end for

                adicionar el nodo niwj como hijo del nodo subproceso
                        nsi

                diwj ← CREATEDIAGNOSISINFORMATION(ws, li, V)

                adicionar diwj al perfil de diagnóstico.

            end if

        end for

    end if
```

```
    end while
end Procedimiento
```

La identificación de los patrones de control de flujo se realiza a partir de la combinación de comportamientos presentes en el registro de eventos y comportamientos asumidos bajo el umbral de ausencia de información. Para encontrar la combinación de comportamientos que puede conducir a la descomposición de un subproceso por un determinado patrón de control de flujo se utiliza el procedimiento que se expone a continuación (PÉREZ ALFONSO, 2014).

Algoritmo Búsqueda de variantes

Entrada: patrón de control de flujo (**wj**), comportamiento del proceso de la sección del registro de eventos **li(βli)**, umbrales de ruido y completitud (**TP**)

Salida: variante de descomposición en subprocesos a través del patrón **wj**

```
Procedimiento FINDVARIANTS(wj, βli, TP)
    βwjli ← SORTBYFREQUENCY(βli[wj]) //Ordenando comportamiento en orden
            descendente
    τwj ← TP[wj] //Umbrales para el patrón wj
    Ywjsi ← CREATEFIRSTDECOMPOSITIONPROPOSAL(wj, βwjli, τwj)
    if Ywjsi = ∅ then //βwjli no es suficiente para crear una propuesta de
            descomposición bajo τwj
        return ∅
    end if
    ψ1 ← CREATESEARCHNODE(βwjli, Ywjsi)
    Ψ ← {ψ1} //Cola con prioridad de nodos abiertos de acuerdo a c
    while ψn ← extract first node on Ψ do
        if ψn is goal & βwjli = ∅ then
            g ← ψn
        else if ψn is not closed then
```

```
                Ψn ← SEARCHN EIGHBORS(wj, ψn, βwjli, τwj)
                Ψ ← Ψ + Ψn //Adicionar vecinos como nodos abiertos
            end if
        end while
        if g is not null then
            Ywjsi ← descomposición potencial en g
            return CREATENEWSUB-PROCESSES(wj, Ywjsi, βli)
        else
            return Ø
        end if
end Procedimiento
```

La búsqueda de variantes está basada en la Búsqueda de costo uniforme (Russell et al., 1995). Este método de búsqueda expande el nodo que posea el menor costo de camino. En caso de que los costos de todos los pasos de un nodo a otro sean iguales, es idéntico a una búsqueda a lo ancho. Si el costo de cada paso es menor o igual que una constante ε, se garantiza la completitud y optimalidad del método. La complejidad temporal y espacial del peor caso de este método de búsqueda pueden ser descritas por la ecuación $r^{1+C/\varepsilon}$, donde r es el factor de ramificación del árbol y C es el costo del camino de la solución óptima (PÉREZ ALFONSO, 2014).

1.3.3.4 Generación del perfil de diagnóstico

Esta es la última etapa que se ejecuta dentro de la técnica. El algoritmo implementado en esta etapa recorre el árbol de búsqueda generado y recopila los comportamientos considerados como ruido y ausencia de información durante la identificación de los patrones. A partir de estos comportamientos asumidos o descartados se establecen la estimación de aptitud y precisión. El perfil se conforma con la información de diagnóstico de cada nodo de tipo patrón en el Árbol de Variantes obtenido.

1.4Tecnologías para el desarrollo

A continuación se mencionan las herramientas y tecnologías utilizadas durante el desarrollo de la presente investigación.

1.4.1 Marco de trabajo para la minería de proceso ProM

ProM es un marco de trabajo para la minería de proceso. Esta es una herramienta de código abierto y puede ser descargada gratis desde www.processmining.org. ProM provee a los usuarios y desarrolladores de una plataforma para la minería de proceso. Contiene complementos para importar y exportar tanto registros de eventos como modelos de proceso, conversión de modelos, técnicas de descubrimiento de proceso y de análisis, entre otros. Actualmente se ha convertido en la herramienta de referencia para la minería de proceso (AALST., 2011). Para la presente investigación se utilizó la versión 6.4.

1.4.2 Lenguaje de programación Java

Java es un lenguaje de programación orientado a objetos y de alto nivel. Java es multiplataforma, lo que significa que el código compilado puede ser ejecutado en cualquier dispositivo y/o sistema operativo. Está considerado en la actualidad entre los lenguajes de programación más populares entre los desarrolladores por la comunidad de desarrollo TIOBE (www.tiobe.com), por la variedad de funcionalidades que permite: aplicaciones de escritorio, para dispositivos móviles, para la web, etcétera. Java fue desarrollado originalmente por James Gosling de Sun Microsystems y publicado en 1995. La sintaxis en gran medida está inspirada en lenguajes como C y C++. El lenguaje al que son compiladas las aplicaciones escritas en Java se llama *bytecode*, el cual puede ejecutarse en cualquier máquina virtual de Java (Rajkumar Buyya, 2009). Se utilizó la versión 8 con la actualización 20.

1.4.3 Entorno de desarrollo integrado NetBeans

NetBeans es un entorno de desarrollo integrado que permite el desarrollo eficiente 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. Permite gestionar de manera eficiente el proyecto con varias vistas para los datos.

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 (Oracle, 2015).

1.4.4 Herramienta CASE Visual Paradigm

Visual Paradigm es una herramienta de Ingeniería de Software Asistida por Computadora (CASE por sus siglas en inglés). Utiliza el lenguaje de modelado UML. Esta herramienta permite la generación, diseño y gestión de los artefactos generados durante el proceso de desarrollo de software. Posee una fácil integración con entornos de desarrollo integrado como NetBeans, Eclipse y Visual Studio. Permite

generar diagramas de clase a partir del código fuente y viceversa, con soporte para C, C++, PHP y Java entre otros. En el proceso de desarrollo de la solución se utilizó la versión 8.0.

1.4.5 Metodología de desarrollo de software empleada en la solución

El proceso de desarrollo de software es una tarea compleja, la cual requiere un alto nivel de organización y detalle. Para la obtención de software con una buena calidad se necesita aplicar una metodología que sea capaz de llevar a cabo el control del proceso. Las metodologías de desarrollo de software surgen ante la necesidad de emplear una serie de técnicas, procedimientos y herramientas para obtener un producto de calidad.

Dada la necesidad de obtener un producto en un período corto de tiempo y la composición del grupo de desarrollo se elige utilizar un enfoque ágil para el desarrollo de la solución.

Programación eXtrema, eXtreme Programming, o simplemente XP; es una metodología de desarrollo de software ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. XP se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y el coraje para enfrentar los cambios (Canós, 2008).

1.5 Conclusiones parciales.

La minería de proceso es la disciplina científica que permite descubrir, monitorear y mejorar los procesos reales, a través de la extracción de conocimiento de los registros de eventos. Las técnicas de descubrimiento de proceso obtienen un modelo de proceso a partir de un registro de eventos. El descubrimiento de proceso es la tarea de la minería de proceso con mayores retos y dificultades. Esta es la etapa a la que se le ha dedicado mayor cantidad de tiempo y esfuerzo.

Varios criterios permiten evaluar la calidad de los modelos obtenidos, entre otros: la aptitud, la correctitud, precisión y generalización, así como el tiempo de ejecución. Características de los registros de eventos como el ruido, y la ausencia de información provocan que los algoritmos de descubrimiento obtengan modelos que no se ajustan a los registros de eventos, o que son muy complejos.

El ruido y la ausencia de información son las características de los registros de eventos que mayor dificultad presentan a las técnicas de descubrimiento. Varias aproximaciones se han propuesto para tratar el ruido y las actividades infrecuentes.

El análisis de las técnicas expuestas permitió identificar un grupo de fortalezas y debilidades que presentan las mismas. El algoritmo de CCM es muy efectivo en el control del impacto del ruido en los registros de eventos. La técnica de minería de variantes presenta una gran debilidad en la

descomposición de registros de eventos con alto número de sucesiones directas entre las actividades, debido a que el costo computacional aumenta significativamente.

Capítulo 2

2.1 Introducción

En este capítulo se presenta la solución desarrollada para tratar con el ruido y la ausencia de información dentro de la técnica Minería de variantes. La solución aprovecha las ventajas del algoritmo de competición del CCM en la extracción del comportamiento del proceso y el descubrimiento de los patrones de control de flujo. La idea principal de la solución es combinar ambas técnicas para suplir las desventajas de una con las fortalezas de la otra. Con esto se soluciona la explosión de estados y el crecimiento acelerado del espacio de búsqueda de la técnica Minería de Variantes ante registros de eventos complejos. Por su parte el algoritmo de competición de CCM es modificado, haciendo que compita en cada descomposición un único patrón de control de flujo. De esta manera se puede lograr diferentes descomposiciones para cada subproceso analizado. La descomposición, por cada patrón, obtiene dos conjuntos disjuntos de actividades $A1\ y\ A2 \mid A1 \cup A2 = A$(Conjunto de todas las actividades del proceso) $A1 \cap A2 = \emptyset$, que representan las ramas de un árbol de variantes. En el proceso de integración entre las dos técnicas, se modifican las fases de Extracción del comportamiento y de Obtención de variantes de descomposición de la Minería de variantes y el algoritmo de competición CCM.

Se realiza una descripción de la solución propuesta, teniendo en cuenta que dicha solución no es en sí una aplicación informática, pero se hace necesario llevar a cabo la documentación de este proceso de desarrollo. Debido a la metodología seleccionada se crean las **historias de usuarios** y el **plan de iteraciones,** se definen los **requisitos funcionales** y se muestra el **diagrama de componentes** de la solución**, las tarjetas CRC** y el **diagrama de clases del diseño.**

2.2 Extracción del comportamiento

En la etapa de extracción del comportamiento de la técnica Minería de variantes se analiza el registro de eventos en busca de relaciones existentes entre las actividades. El algoritmo CCM presenta algunas ventajas en lo que respecta a la extracción de las sucesiones indirectas en la fase de extracción del comportamiento, ya que se centra en reconocer las relaciones globales entre las actividades y no en la relación local, lo que representa una disminución del espacio de búsqueda del algoritmo. La mejora de la técnica radica en que se reduce la explosión de estados en registros de eventos complejos a partir de la extracción de relaciones globales entre las actividades. Nuevos comportamientos son extraídos,

para garantizar que el algoritmo de descomposición sea capaz de descubrir los diferentes patrones de control de flujo. Un ejemplo es la relación global que exista entre cualquier par de actividades: en cuántas trazas un evento x ocurre antes del evento y. Así como información sobre la ocurrencia de un evento. Por ejemplo: cuantas veces aparece en el registro de eventos la actividad x, cuantas trazas comienzan con la actividad x, etcétera.

Definición 1: Sea $L \subseteq A^*$ un registro de eventos, Λ el conjunto de todas las trazas de L y $x \in A$ una actividad, entonces:

1- $Once(x) = \{\lambda \in \Lambda \mid \exists i\ \lambda(i) = x\}$ Todas las trazas del registro de eventos en donde aparece x al menos una vez.

2- $Sum(x) = \{(\lambda, l) \mid \lambda \in \Lambda\ \lambda(l) = x\}$ Todas las apariciones de x en el registro de eventos.

3- $Start(x) = \{\lambda \in \Lambda \mid \lambda(0) = x\}$ Todas las trazas que comienzan con la actividad x.

Sea $x, y \in A$ entonces:

1- $x > y$ si y solo si existe una traza $\lambda \in \Lambda$ y existe un par i, j, $i < j$ tales que $\lambda(i) = x$ y $\lambda(j) = y$ y $\forall_{k \in \{1,2,\dots,j-1\}} \lambda(k) \neq y$. x aparece antes que la primera y.

2- $x >> y$ si y solo si existe una traza $\lambda \in \Lambda$ y existe un par i, j; $i < j$ tales que $\lambda(i) = x$ y $\lambda(j) = y$. x aparece antes que y.

3- $|x > y|$ el número de ocurrencias de $x > y$ en L.

4- $|x >> y|$ el número de ocurrencias de $x >> y$ en L.

Si se toma como ejemplo el conjunto de actividades $A = \{a, b, c, d\}$ y el registro de eventos $L1 = \{\langle a, b, c, d\rangle^2, \langle b, a, c, b, d\rangle^1\}$ se tiene:

- $|Once(b)| = 3$. (Dos veces en $\langle a, b, c, d\rangle^2$ y una vez en $\langle b, a, c, b, d\rangle^1$).
- $|Sum(b)| = 4$. (Dos veces en $\langle a, b, c, d\rangle^2$ y dos en $\langle b, a, c, b, d\rangle^1$).
- $|Start(b)| = 1$. (solo $\langle b, a, c, b, d\rangle^1$ comienza con b).
- $|a > b| = 2$. (solo en $\langle a, b, c, d\rangle^2$ a aparece antes que la primera b).
- $|a >> b| = 3$. (Dos veces en $\langle a, b, c, d\rangle^2$ y una vez en $\langle b, a, c, b, d\rangle^1$).

Durante la ejecución del algoritmo, varias matrices son creadas. Primero una matriz general, abarcando todas las actividades del proceso. Mientras el algoritmo prosigue en su enfoque divide y vencerás, nuevos subconjuntos de actividades son obtenidos, para los cuales una nueva matriz es creada. Los valores de esta matriz están relacionados con el comportamiento que presenten las actividades en el registro de eventos.

Definición 2. Sea $L \subseteq A^*$ un registro de eventos sobre el conjunto de actividades A, $|\Lambda|$ el número de trazas presentes en L y $x \in A$ una actividad, se tiene:

- Los valores de $ocurrencia\ única\ Oon(x)\ y\ ocurrencia\ total\ Oov(x)$ se calculan como sigue:

$$Oon(x) = \frac{|Once(x)|}{|\Lambda|} \qquad Oov(x) = \frac{|Sum(x)|}{|\Lambda|}$$

- El valor $primer\ elemento\ Fel(x)$ se calcula con la siguiente ecuación:

$$Fel(x) = \frac{|Start(x)|}{|\Lambda|}$$

Sean $x, y \in A$ un par de actividades en L:

- El valor $ocurre\ antes\ de\ x \rhd\rhd y$ y $ocurre\ antes\ que\ el\ primer\ x \rhd y$ se calculan como sigue:

$$x \rhd\rhd y = \frac{|x>>y|}{|\Lambda|} \qquad x \rhd y = \frac{|x>y|}{|\Lambda|}$$

Todos los valores en $Oon(x)$, $Fel(x)$, $x \rhd\rhd y$ y $x \rhd y$ serán $\geq 0\ y \leq 1$ dado que este tipo de comportamiento ocurre a lo sumo una vez por traza, mientras que el valor de $Fel(x)$ se hace mayor que 1 cuando x ocurre como promedio más de una vez por traza.

Con la ayuda de estos valores absolutos, la matriz puede ser calculada en relación con el total de trazas. Posteriormente, basado en estos valores, el algoritmo evalúa restricciones para los subconjuntos de actividades que permiten identificar la evidencia de un patrón de flujo determinado. Para el cálculo de los valores en matriz se tiene:

La matriz completa consiste en los valores $Oon, Oov, Fel, x \rhd\rhd y$ y $x \rhd y$. Por ejemplo, para el registro $L1$ la matriz $M = \{Oon, Oov, Fel, x \rhd\rhd y, x \rhd y\}$ sería:

$$L_1 = \{[b,a]^4, [a,b,d,e]^5, [b,a,e,d]^4, [b,a,c,a,b,c,b,a,d,e,e,d]^6, [g,g]^2, [f,h]^3, [f,f,h,f,g,h,g,f,h]^8, [g,h,f]^2\}$$

$$Oon(x): \begin{array}{cccccccc} a & b & c & d & e & f & g & h \\ 0.56 & 0.56 & 0.18 & 0.44 & 0.44 & 0.38 & 0.35 & 0.44 \end{array}$$

$$Oov(x): \begin{array}{cccccccc} a & b & c & d & e & f & g & h \\ 0.91 & 0.91 & 0.35 & 0.62 & 0.62 & 1.09 & 0.65 & 0.85 \end{array}$$

$$Fel(x): \begin{array}{cccccccc} a & b & c & d & e & f & g & h \\ 0.15 & 0.41 & 0.00 & 0.00 & 0.00 & 0.32 & 0.12 & 0.00 \end{array}$$

$x \rhd\rhd y$:

	a	b	c	d	e	f	g	h
a	0.18	0.32	0.18	0.44	0.44	0	0	0
b	0.41	0.18	0.18	0.44	0.44	0	0	0
c	0.18	0.18	0.18	0.18	0.18	0	0	0
d	0	0	0	0.18	0.32	0	0	0
e	0	0	0	0.29	0.18	0	0	0
f	0	0	0	0	0	0.24	0.24	0.32
g	0	0	0	0	0	0.29	0.29	0.29
h	0	0	0	0	0	0.29	0.24	0.24

$x \rhd y$:

	a	b	c	d	e	f	g	h
a	0	0.15	0.18	0.44	0.44	0	0	0
b	0.41	0	0.18	0.44	0.44	0	0	0
c	0	0	0	0.18	0.18	0	0	0
d	0	0	0	0	0.32	0	0	0
e	0	0	0	0.18	0	0	0	0
f	0	0	0	0	0	0	0.24	0.32
g	0	0	0	0	0	0.06	0	0.06
h	0	0	0	0	0	0.06	0.24	0

2.3 Obtención de variantes de descomposición

En esta fase de la Minería de variantes se obtienen varias descomposiciones para una misma sección del proceso. Con la incorporación del algoritmo de competición CCM es necesario realizar modificaciones al mismo para que por cada descomposición compita un solo patrón de control de flujo. De esta manera, al separar los patrones, por cada descomposición es posible obtener más de una variante, sacando provecho de la mejora en tiempo computacional del CCM.

La entrada de este algoritmo consiste en las trazas del registro de eventos con su frecuencia. A partir del análisis de las trazas se extrae el comportamiento necesario para construir la matriz compuesta de los valores $Oon, Oov, Fel, x \rhd\rhd y$ y $x \rhd y$. A continuación se expone el algoritmo implementado.

```
metodo decompose

extractBehavior();

initMatrixes();
```

```
if actividades.size == 1
  determineActivityType();
else {
  q ← new PriorityQueue<QueueTuple>();
  SeqEmptyCase ← new QueueTuple({},{},{process.activities, 0});
  q.add(SeqEmptyCase);
  while(true){
   c ← q.poll();
   if (c.aLeft.size == 0)
      return {c.aFirst, c.aSecond };
   else {
     if (c.aSecond.size > 0 or c.aLeft.size > 0) {
       x ← c.aLeft.poll();
       aNew ← c.aFirst;
       aNew.add(x);
       pNew ← 0.0;
       foreach(y : c.aSecond)
         pNew += evaluate(x, y);
       if (pNew > 0)
         pNew ← pNew / aSecond.size + c.penalty;
       else
         pNew ← c.penalty;
```

```
    q.add(new QueueTuple(aNew, c.aSecond, c.aLeft, pNew));
  }
  if (c.aFirst.size > 0 or c.aLeft.size > 0) {
    x ← c.aLeft.poll();
    aNew ← c.aSecond;
    aNew.add(x);
    pNew ← 0.0;
    foreach(y : c.aFirst)
      pNew += evaluate(y, x);
    if (pNew > 0)
      pNew ← pNew / aSecond.size + c.penalty;
    else
      pNew ← c.penalty;
    q.add(new QueueTuple(c.aFirst, aNew, c.aLeft, pNew));
    }
```

En la cola con prioridad q se almacenan tuplas que tienen la forma $T = \{aFirst, aSecond, aLeft, penalty\}$ donde $aFirst$ y $aSecond$ son los conjuntos en que se descompone el proceso, $aLeft$ contiene las actividades del proceso que aún no han sido asignadas a $AFirst$ o $aSecond$ y un valor $penalty$, que representa la penalidad de evaluar las restricciones para descubrir los patrones de control de flujo sobre los subconjuntos de actividades $aFirst$ y $aSecond$.

Las restricciones a evaluar sobre los conjuntos de actividades son:

XOR $x \rhd\!\rhd y \cong 0$ $\quad y \rhd\!\rhd x \cong 0$

Paralelismo $x \rhd y \ncong 0$ $\quad y \rhd x \ncong 0$ $\quad coc(x,y) \cong oc(x,y)$

Secuencia $x \rhd\rhd y \not\cong 0$ Lazo $rep(x) \not\cong 0$

$y \rhd\rhd x \cong 0$ $rep(y) \not\cong 0$

$x \rhd\rhd y \cong x \rhd y$

Los valores $oc\ (ocurrencia)$, $coc\ (ocurrencia\ combinada)$ $y\ rep\ (repeticion)$ son calculados mediante las formulas:

$$oc(x,y) = x \rhd y + y \rhd x \qquad coc(x,y) = Oon(x) * Oon(y)$$

$$rep(x) = \frac{Oov(x) - Oon(x)}{Oov(x)}$$

El algoritmo es capaz de asimilar el ruido y el comportamiento poco frecuente, estableciendo umbrales para los mismos, además permite descubrir en el registro de eventos los patrones de control de flujo Secuencia, Selección, Ciclo y Paralelismo, evaluando restricciones sobre el comportamiento apreciado en las trazas.

2.4 Historias de Usuarios

Las Historias de Usuario (HU) son los artefactos generados en la metodología XP para especificar los requisitos del software. En estas se describe brevemente las características con que debe cumplir el sistema a desarrollar. Los requisitos para el software se agrupan en las siguientes 5 HU.

Tabla 1: Historias de Usuario "Extracción del comportamiento".

Historia de Usuario	
Numero: HU01	Nombre de la Historia de Usuario: Extracción del comportamiento
Modificación de la Historia de Usuario: Ninguna	
Usuario: Alfredo Aguilera Miranda	Iteración: 1
Prioridad en negocio: Alta	Puntos estimados: 2
Riesgo en desarrollo: Medio	Puntos reales: 2
Descripción: Se extraen del registro de eventos los comportamientos siguientes: • Ocurrencia única • Ocurrencia total • Actividades que inician trazas • X ocurre antes que Y	

• X ocurre antes que la primera Y
Observaciones

Tabla 2: Historias de Usuario "Generación de las matrices de comportamiento"

Historia de Usuario		
Numero: HU02	Nombre de la Historia de Usuario: Generación de las matrices de comportamiento	
Modificación de la Historia de Usuario: Ninguna		
Usuario: Alfredo Aguilera Miranda		Iteración: 1
Prioridad en negocio: Alta		Puntos estimados: 1
Riesgo en desarrollo: Medio		Puntos reales: 1
Descripción: Se generan las matrices de comportamiento que contiene los siguientes valores: - Oon(x) ocurrencia única - Oov(x) ocurrencia total - Fel(x) primer elemento - x >> y x ocurre antes que y - x > y x ocurre antes que la primera y		
Observaciones		

Tabla 3: Historias de Usuario "Descomposición de las actividades de un proceso"

Historia de Usuario		
Numero: HU03	Nombre de la Historia de Usuario: Descomposición de las actividades de un proceso	
Modificación de la Historia de Usuario: Ninguna		
Usuario: Alfredo Aguilera Miranda		Iteración: 2
Prioridad en negocio: Alta		Puntos estimados: 1
Riesgo en desarrollo: Medio		Puntos reales: 1
Descripción: Se crean las posibles descomposiciones del conjunto de actividades de un proceso		
Observaciones		

Tabla 4: Historias de Usuario "Evaluación de restricciones sobre los subconjuntos obtenidos"

Historia de Usuario	
Numero: HU04	Nombre de la Historia de Usuario: Evaluación de restricciones sobre los subconjuntos obtenidos
Modificación de la Historia de Usuario: Ninguna	
Usuario: Alfredo Aguilera Miranda	Iteración: 2
Prioridad en negocio: Alta	Puntos estimados: 1
Riesgo en desarrollo: Medio	Puntos reales: 1
Descripción: A partir de las descomposiciones obtenidas para un subproceso, se evalúan un conjunto de restricciones sobre los mismos. De esta forma se determina el ajuste de los subconjuntos de actividades hacia el patrón de control de flujo Secuencia.	
Observaciones	

Tabla 5: Historias de Usuario "Competición de las variantes de descomposición"

Historia de Usuario	
Numero: HU05	Nombre de la Historia de Usuario: Competición de las variantes de descomposición
Modificación de la Historia de Usuario: Ninguna	
Usuario: Alfredo Aguilera Miranda	Iteración: 2
Prioridad en negocio: Alta	Puntos estimados: 1
Riesgo en desarrollo: Medio	Puntos reales: 1
Descripción: Se determina cual descomposición es la que mejor se ajusta a un patrón de control de flujo determinado.	
Observaciones	

2.5 Plan de iteraciones

Se planifica que el proceso de desarrollo se realizará en dos iteraciones. La siguiente tabla muestra la distribución de las HU que se implementarán por cada iteración.

Tabla 6: Plan de iteraciones

Iteración	**HU a implementar**	**Duración(semanas)**
1	Extracción del comportamiento	1

1	Generación de las matrices de comportamiento	1
2	Descomposición de las actividades de un proceso	1
2	Evaluación de restricciones sobre los subconjuntos obtenidos	1
2	Competición de las variantes de descomposición	1
Total		5

2.6 Requisitos funcionales

A partir de las Historias de Usuario definidas, se especifican los requisitos funcionales con que deberá cumplir el sistema. La siguiente tabla muestra un resumen de los requisitos funcionales definidos.

Tabla 7: Requisitos funcionales

No.	Requisito funcional
1	Extracción del comportamiento ocurrencia única
2	Extracción del comportamiento ocurrencia total
3	Extracción del comportamiento primer elemento
4	Extracción del comportamiento x ocurre antes que y
5	Extracción del comportamiento x ocurre antes que la primera y
6	Generación de las matrices de comportamiento
7	Descomposición de las actividades de un proceso
8	Evaluación de restricciones sobre los subconjuntos obtenidos
9	Competición de las variantes de descomposición

Fase de desarrollo

La fase de desarrollo tiene como objetivo codificar los requisitos funcionales definidos para el sistema. La utilización de estándares de codificación permite garantizar la legibilidad y entendibilidad del código generado, garantizando así la calidad del software. En el presente apartado se presentan los estándares de codificación utilizados durante el proceso de desarrollo así como los artefactos generados por la metodología XP para garantizar una mejor documentación del proyecto.

2.7 Estándares de codificación

Nomencladores

Para los nombres de clases: Los nombres de clases comenzarán con letra mayúscula y de ser nombres compuestos cada letra inicial de cada palabra se escribirá con letra mayúscula. Deberán expresar de manera precisa y sencilla la función que desempeñan.

Para los nombres de variables y métodos: se comenzará cada nombre de variable o método con letra inicial minúscula, de ser nombres compuestos, se escribirá cada nueva palabra con letra inicial mayúscula. Los nombres de variables deberán significar el contenido de las mismas así como los nombres de métodos deberán expresar la función que realizan.

Comentarios

Para la documentación del código se utilizaron comentarios que explican en cada momento la función de un bloque, asignación o cualquier porción de código. El uso de comentarios es una buena práctica cuando se escribe código. Permite a los desarrolladores entender de una manera más fácil y rápida el código, permitiendo que se reduzca el esfuerzo del mantenimiento del software.

```
//ejemplo de un comentario de línea
/*
Ejemplo de un bloque de
Comentarios
*/
```

2.8 Diagrama de componentes

En el paquete ***behavior*** se encuentran los componentes encargados de analizar el registro de eventos y extraer los comportamientos presentes en el mismo. El componente ***behaviorMinerByCCM*** es el encargado de extraer las relaciones **ocurrencia única**, **ocurrencia total**, **primer elemento de traza**, **x aparece antes que y**, y **x aparece antes que la primera y**. El paquete ***patternsByCCM*** contiene los componentes necesarios para descomponer el proceso en patrones de control de flujo. En este paquete se encuentra el componente ***Pattern Decomposition Manager By CCM***, que lleva a cabo la descomposición del proceso en conjuntos de actividades, teniendo en cuenta el patrón de control de flujo al que mejor se ajustan. En este último paquete se encuentran además los componentes encargados de evaluar las restricciones definidas para los patrones de control de flujo sobre los conjuntos de actividades obtenidas

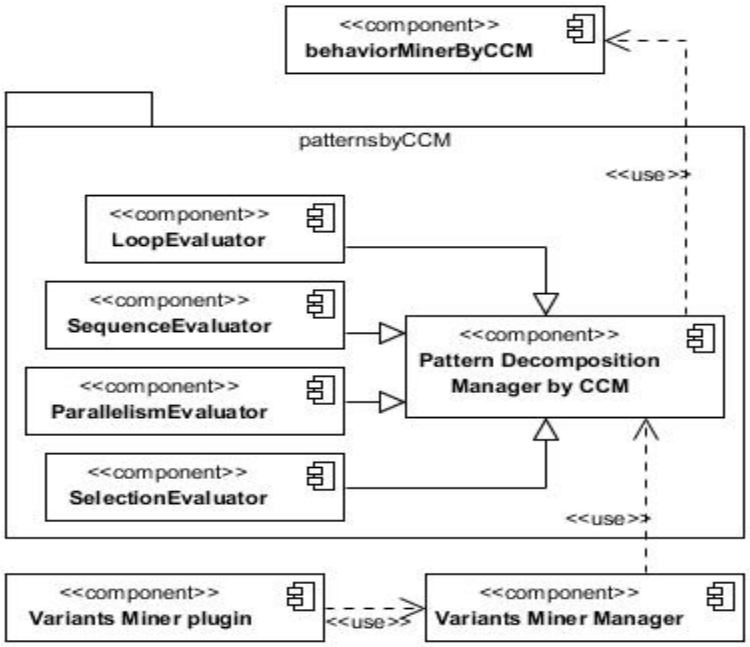


Figura 7: Diagrama de componentes

2.9 Tarjetas Clase-Responsabilidad-Colaboración (CRC).

Las tarjetas CRC, son el artefacto generado por la tecnología XP para describir las clases del diseño. Estas explican las responsabilidades que tiene una clase, así como las clases con que colabora la misma. A continuación se muestran las tarjetas CRC para las clases CCMTuple, PatternManagerByCCM, BehaviorMinerByCCM y SequenceEvaluator.

Clase **CCMTuple**	
Responsabilidades	Colaboración
• Almacenar los conjuntos de descomposición de los procesos. • Son las entidades que compiten en el algoritmo de descomposición. • Establecer el orden de prioridad de una descomposición según la penalidad de evaluar las restricciones sobre los patrones.	PatternManagerByCCM

Clase **BehaviorMinerByCCM**	
Responsabilidades	Colaboración
• Extraer el comportamiento expresado en el registro de eventos.	PatternManagerByCCM

Clase **PatternManagerByCCM**	
Responsabilidades	Colaboración
• Invocar la funcionalidad extraerComportamiento() de la clase BehaviorMinerByCCM. • Generar las matrices de comportamiento. • Obtener las variantes de descomposición. • Devolver las descomposiciones para un subproceso analizado.	CCMTuple, BehaviorMinerByCCM, SequenceEvaluator, SelectionEvaluator, XOREvaluator, ParallelismEvaluator

2.10 Diagrama de clases del diseño

En el diagrama de clases del diseño se muestra la estructura del sistema en cuanto a las relaciones que presentas las clases del diseño implementadas. Las clase BehaviorMinerByCCM contiene el método extractBehavior(), encargado de procesar las trazas del registro de eventos y extraer los comportamientos presentes en el mismo. La funcionalidad de la clase PatternManagerByCCM es la de calcular las matrices de comportamiento para el subproceso analizado mediante el método initMatrixes(). El método decompose() descompone las actividades del proceso en dos subconjuntos disjuntos. En este método esta implementado el algoritmo de competición de la técnica CCM, el cual determina qué par de conjuntos de actividades, así como qué patrón de control de flujo resulta ganador de la descomposición. Las clases ParallelismEvaluator, XOREvaluator, SequenceEvaluator, y LoopEvaluator heredan de la PatternManagerByCCM y tienen la funcionalidad de evaluar las restricciones específicas de cada patrón de control de flujo sobre los subconjuntos de actividades obtenidos. Al separar las evaluaciones de cada patrón de control de flujo, es posible obtener varias descomposiciones para un mismo subproceso analizado.

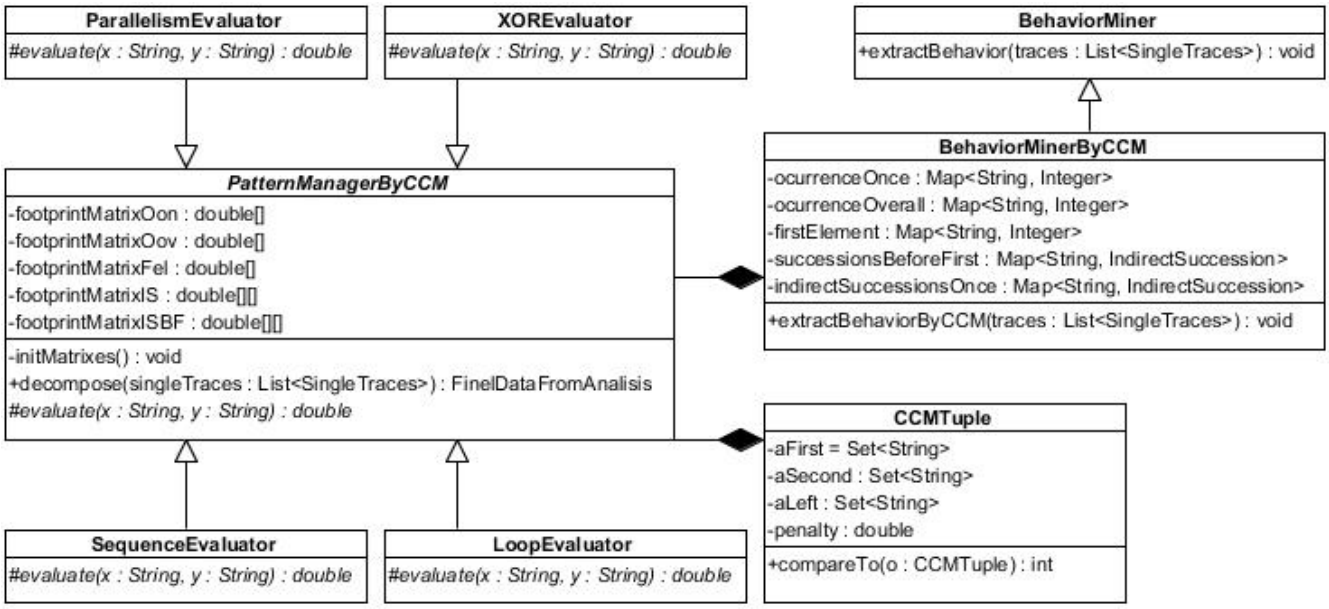


Figura 8. Diagrama de clases del diseño.

2.11 Conclusiones parciales

La integración entre Minería de Variantes y Constructs Competition Miner (CCM) permitió obtener un algoritmo de descubrimiento de proceso que combina las fortalezas de ambas técnicas. Este algoritmo permite el diagnóstico de variantes de proceso en registros de eventos con presencia de ruido y ausencia de información.

Capítulo 3

3.1 Introducción

Para la validación de la solución propuesta se realizaron pruebas experimentales para comparar los resultados de otras técnicas de minería de proceso con el resultado del algoritmo implementado. En particular se utilizaron las siguientes técnicas: Inductive Miner (IM), Heuristic Miner (HM) y Minería de Variantes (MV) sin la implementación del algoritmo de competición. El objetivo fundamental de esta comparación es identificar si la calidad los modelos obtenidos por la solución propuesta es mejor o al menos similar a la de los modelos obtenidos por las otras técnicas, en registros de eventos con ruido y ausencia de información. Para cuantificar la calidad de los modelos descubiertos durante las pruebas se utilizaron métricas de calidad establecidas en la literatura. La evaluación de los modelos descubiertos permite a su vez evaluar el desempeño de las técnicas de descubrimiento de proceso. La presente validación se estructura en las siguientes etapas:

Etapa 1. Selección de registros de eventos con presencia de ruido y ausencia de información.

Etapa 2. Aplicación de las técnicas de descubrimiento de proceso elegidas a los registros de eventos seleccionados.

Etapa 3. Evaluación de los modelos de proceso descubiertos a partir del cálculo de métricas de calidad.

3.2 Descripción de la validación

Etapa 1: Para la realización de la evaluación de calidad se seleccionan registros de eventos con características específicas. Un criterio fundamental en esta selección consiste en la presencia de patrones de control de flujo como: XOR o selección exclusiva, AND o paralelismo, Lazos o actividades repetidas y Secuencias, así como combinaciones de estos, como secuencias anidadas en ciclos, etcétera. Otra característica de estos registros de eventos es la presencia en los mismos de ruido o actividades poco frecuentes, y ausencia de información.

Varios registros de eventos fueron seleccionados con diferentes niveles de complejidad, teniendo en cuenta las características deseadas para los mismos:

EX5: (14, 100, 1498) registro de eventos sobre un proceso de revisiones perteneciente a los tutoriales del marco de trabajo ProM (Process MIning, 2015).

REP: (8, 1104, 7733) registro de eventos sobre un proceso de reparaciones de equipos (AALST., 2011).

FL_A (10, 13087, 60849) un registro de eventos extenso sobre un proceso financiero real (BPI Challenge 2012).

Etapa 2: Para selección de las técnicas de descubrimiento para comparar se consideró la similitud respecto al tipo de modelo que descubren y el enfoque del algoritmo de descomposición que emplean en comparación con Minería de Variantes, así como la capacidad de tratar con el ruido y la ausencia de información del registro de eventos. En particular se selecciona IM debido a que el modelo que devuelve es un árbol de proceso, similar al árbol de variantes de MV. Además su algoritmo de descomposición también sigue un enfoque "divide y vencerás" recursivo. HM es muy efectivo cuando el registro de eventos contiene ruido y/o ausencia de información, por lo que fue seleccionado para la realización de las pruebas. Se utilizó además la versión anterior de Minería de Variantes, sin la implementación del algoritmo de competición.

Se ejecutó cada técnica con cada uno de los registros de eventos disponibles. Como resultado de esta etapa se obtuvo un modelo de proceso por cada registro de eventos y por cada técnica. Posteriormente se evaluaron las métricas de calidad sobre estos modelos descubiertos.

Etapa 3: Comprobar la calidad de los modelos descubiertos por cada una de las técnicas a partir de cada registro de eventos seleccionado. La calidad de un modelo se establece a partir de su simplicidad y su capacidad para representar el proceso a partir del cual fue generado el registro de eventos correspondiente (Towards Robust Conformance Checking, 2011). Por tanto la evaluación de la calidad de un modelo se realiza en las siguientes dimensiones:

Aptitud: El modelo descubierto debe permitir el comportamiento observado en el registro de eventos

Precisión: El modelo descubierto no debe permitir comportamientos que no estén relacionados con el que se observa en el registro de eventos

Simplicidad: El modelo descubierto debe ser lo más simple posible.

En esta etapa se compararon los resultados de evaluar estas dimensiones de calidad sobre los modelos de proceso descubiertos. Para los valores de precisión y aptitud, cuanto mayor sea el valor obtenido significa un mejor resultado. Cuanto menor sea el resultado para la dimensión de simplicidad mejor será el resultado obtenido.

3.3 Resultados

A continuación se presentan los resultados obtenidos en la Etapa 3 de la validación. Para facilitar la compresión, los resultados se analizan agrupados por cada uno de los registros de eventos

seleccionados en la Etapa 1. Los resultados obtenidos por la solución propuesta se muestran bajo el nombre VMCCM.

En la siguiente tabla se muestran los resultados obtenidos por las diferentes técnicas seleccionadas sobre el registro de eventos “FLa”. El mismo contiene 10 actividades, con 13087 trazas y un total 60849 ejecuciones de actividades.

Tabla 8 Resultados para el registro de eventos FLa.

	HM	IM	VM	VMCCM
Precisión (Alineación única)	0,58	0,86	0,81	0,9
Aptitud basado en alineación	1	0,99	1	0,99
Simplicidad(Grado promedio de arcos)	2,26	2,43	2,53	2,43
Simplicidad(Conteo de arcos)	52	34	38	34

En la tabla se observa que la técnica Minería de Variantes con el algoritmo de competición CCM (VMCCM) obtiene los mejores resultados en la dimensión **precisión** y empareja con la mejor en **simplicidad (Conteo de arcos).** A continuación se muestran los modelos obtenidos por las diferentes técnicas.

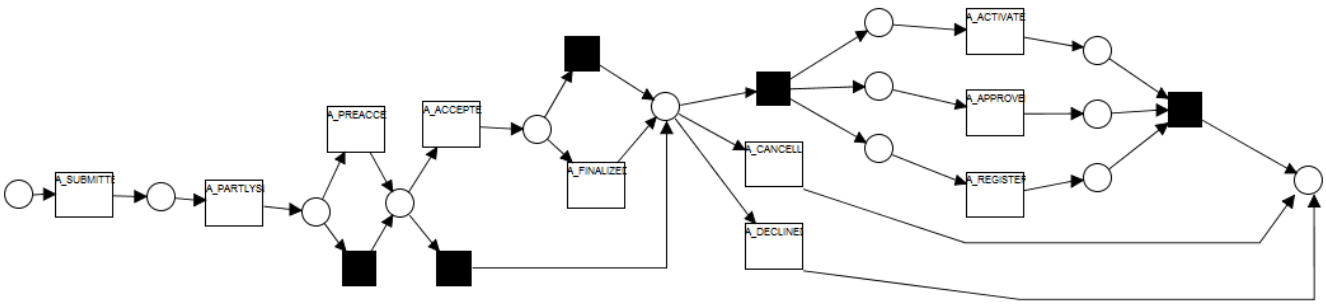


Figura 8a Modelo para el registro FLa de la técnica Minería de Variantes CCM.

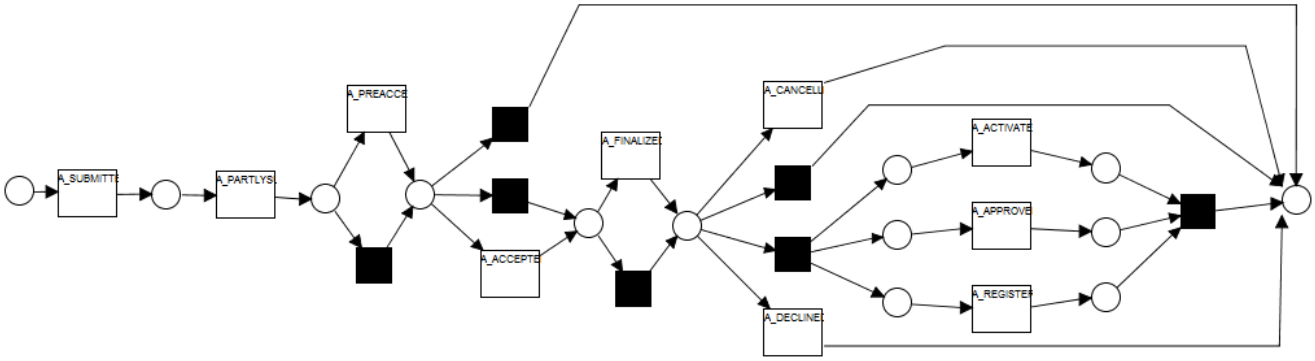


Figura 8b Modelo para el registro FLa de la técnica Mineria de Variantes.

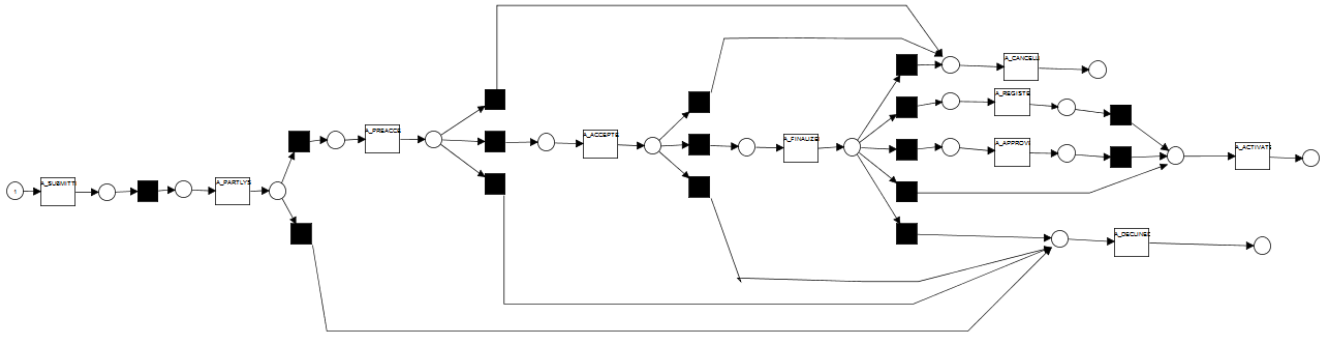


Figura 8c Modelo para el registro FLa de la técnica Heuristic Miner.

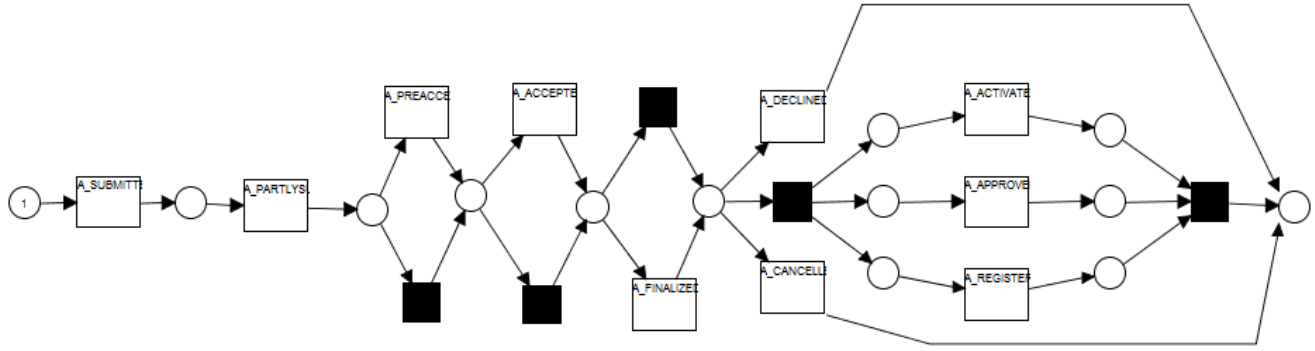


Figura 8d Modelo para el registro FLa de la técnica Inductive Miner.

Como se observa existen diferencias estructurales notables entre los modelos de cada técnica, siendo el modelo de Minería de Variantes con el algoritmo de competición CCM, el más preciso. Además de obtener muy buenos resultados en simplicidad y aptitud.

La siguiente tabla muestra los resultados del experimento realizado sobre el registro de eventos REP, un proceso de reparaciones con 8 actividades, 1104 trazas y un total de 7733 ejecuciones de actividades.

Tabla 9 Resultados para el registro de eventos REP.

	HM	IM	VM	VMCCM
Precisión (Alineación única)	0,912	0,83	0,9	0,9
Aptitud basado en alineación	0,85	0,76	0,99	0,99
Simplicidad(Grado promedio de arcos)	2,29	2,31	2,29	2,28
Simplicidad(Conteo de arcos)	62	30	32	32

La técnica de Minería de Variantes con la implementación del algoritmo de competición CCM obtiene resultados satisfactorios, empatando con el mejor en aptitud y siendo el mejor en simplicidad. A continuación se muestran los modelos obtenidos por las diferentes técnicas para el registro de eventos REP.

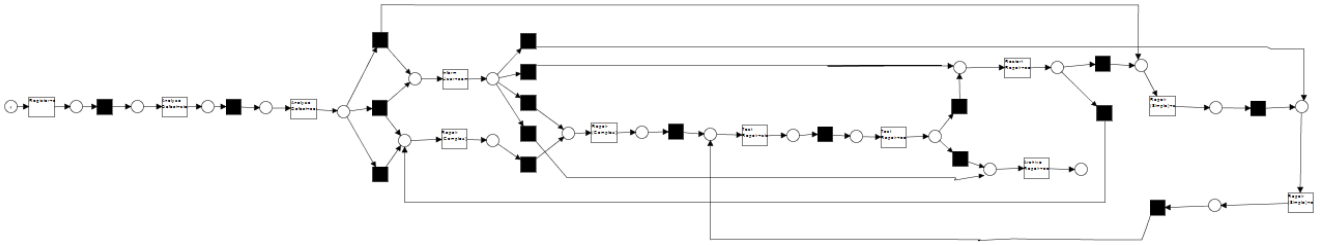


Figura 9a Modelo para el registro REP de la técnica Heuristic Miner.

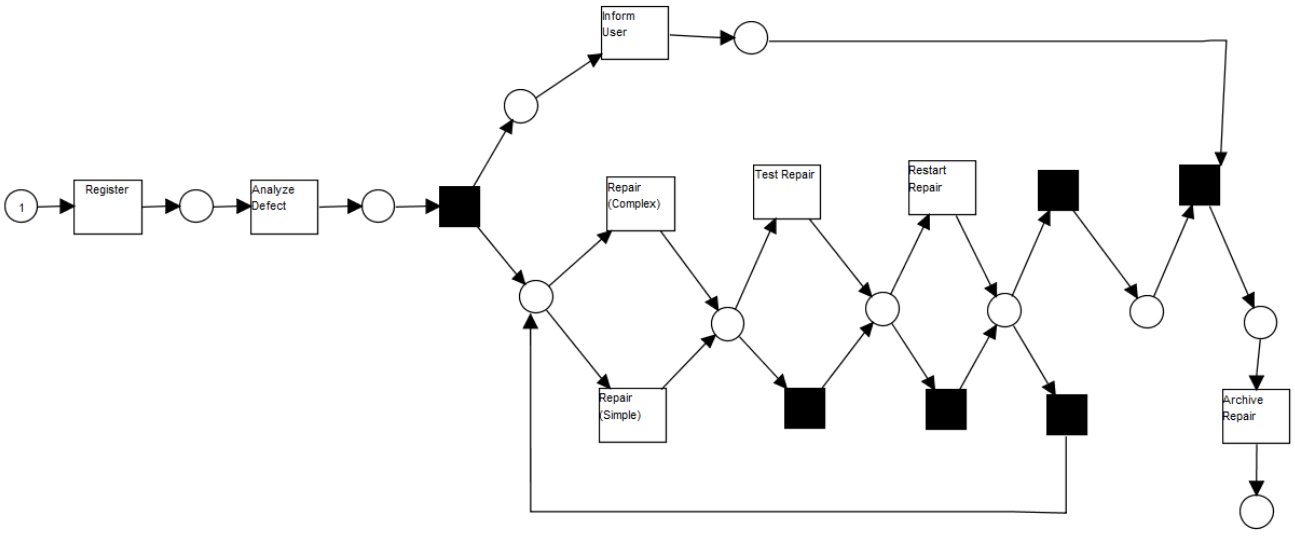


Figura 9b Modelo para el registro REP de la técnica Inductive Miner.

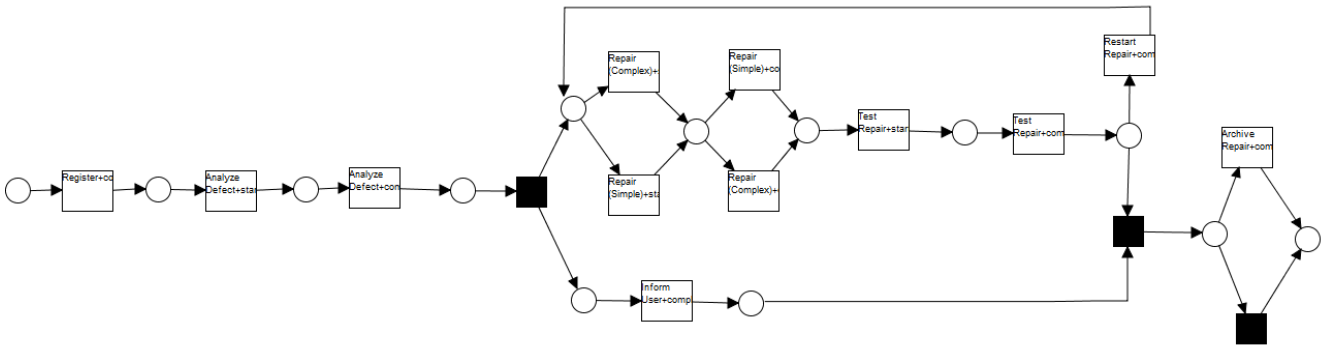


Figura 9c Modelo para el registro REP de la técnica Minería de Variantes.

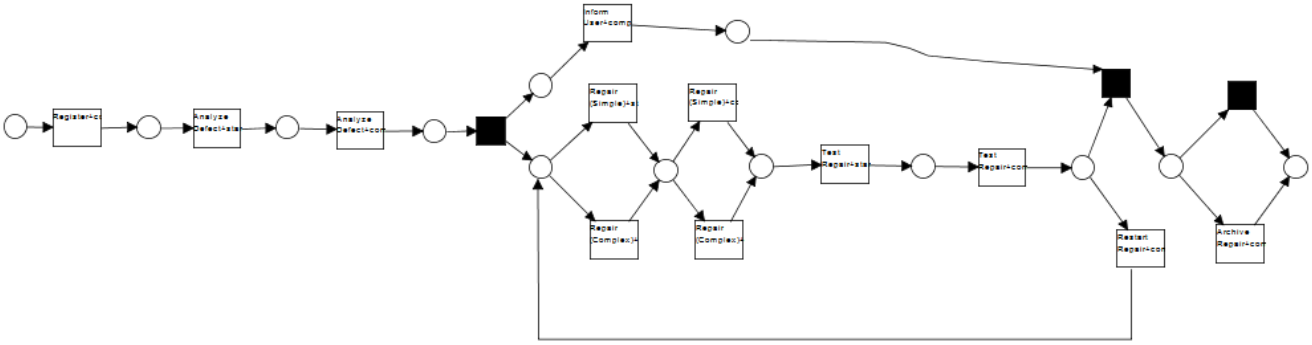


Figura 9d Modelo para el registro REP de la técnica Minería de Variantes CCM.

Tabla 10 Resultados para el registro de eventos EX5.

	HM	IM	VM	VMCCM
Precisión (Alineación única)	0,89	0,72	0,23	0,75
Aptitud basado en alineación	0.84	0.79	0.56	0.82
Simplicidad(Grado promedio de arcos)	2,93	2,49	2,19	2,4
Simplicidad(Conteo de arcos)	132	46	34	54

La Tabla 8 muestra los resultados obtenidos en los modelos descubiertos para el registro de eventos EX5. Es importante destacar la mejoría entre Minería de Variantes sin el algoritmo de competición y la solución implementada. El registro forma parte de los tutoriales para el marco de trabajo para la minería de proceso ProM y puede ser descargado desde www.processminig.org. El mismo cuenta con 14

actividades, 100 trazas y un total de 1498 ejecuciones de actividades. Los modelos descubiertos por las diferentes técnicas se muestran a continuación.

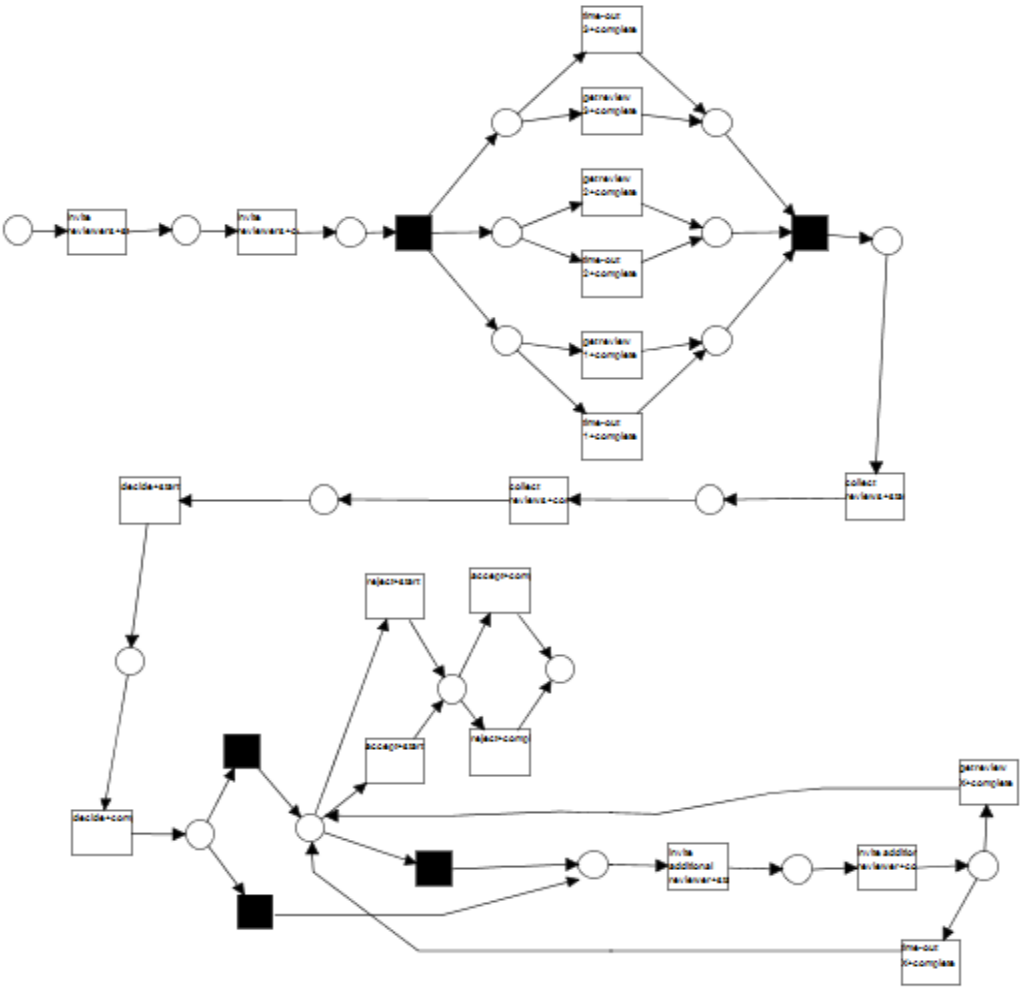


Figura 10a Modelo para el registro EX5 de la técnica Minería de variantes CCM.

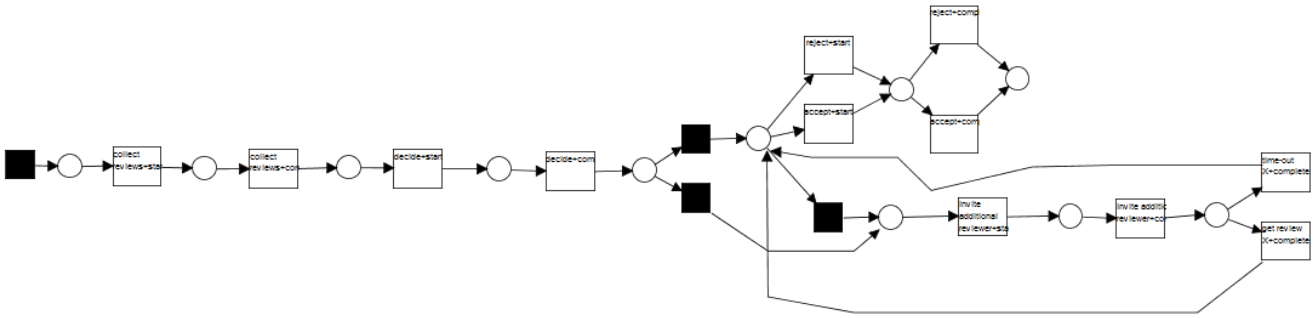


Figura 10b Modelo para el registro EX5 de la técnica Minería de Variantes.

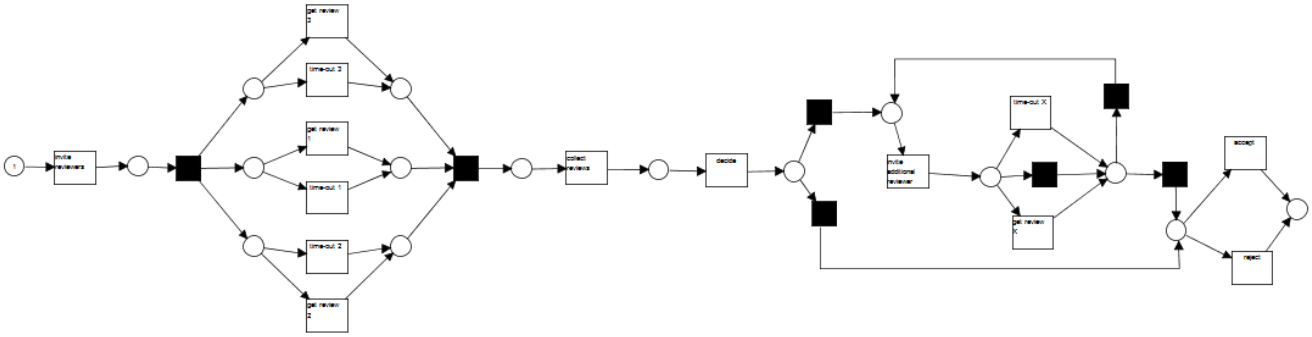


Figura 10c Modelo para el registro EX5 de la técnica Inductive Miner.

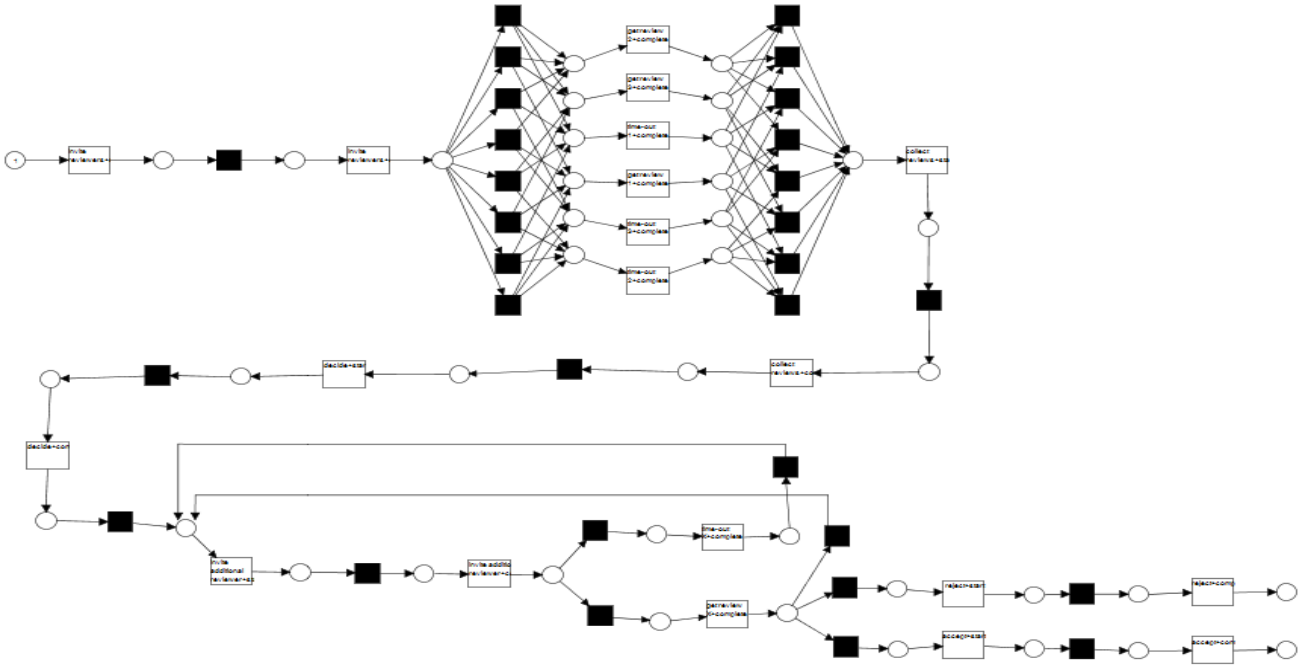


Figura 10d Modelo para el registro EX5 de la técnica Minería de Variantes CCM.

3.4 Conclusiones parciales

La validación de la solución propuesta permitió comprobar una mejoría en cuanto a la calidad de los modelos descubiertos por la técnica Minería de variantes con el algoritmo de competición CCM. Los resultados obtenidos demuestran que la técnica obtiene modelos de calidad ante registros de eventos con presencia de ruido y ausencia de información. La comparación con otras técnicas de minería de

procesos como Heuristic Miner e Inductive Miner muestra que en la mayoría de los casos la solución implementada obtiene modelos que son más precisos y más simples que los obtenidos por otras técnicas.

Conclusiones generales

La investigación realizada, así como el desarrollo de la solución y la validación realizada permitieron obtener las siguientes conclusiones:

- El descubrimiento de proceso es una tarea compleja, con disímiles retos entre los que se encuentran el ruido y la ausencia de información de los registros de eventos.
- La Minería de Variantes es la técnica más completa para el diagnóstico de variantes de proceso, pero la complejidad del algoritmo de descubrimiento que utiliza aumente significativamente ante registros de eventos con alto número de sucesiones directas entre las actividades. El algoritmo Constructs Competition Miner (CCM), aunque no considera la existencia de variantes de proceso, logra balancear el tamaño del espacio de búsqueda y la calidad del modelo descubierto.
- La integración entre Minería de Variantes y Constructs Competition Miner (CCM) permitió obtener un algoritmo de descomposición de proceso que combina las ventajas de ambas técnicas posibilitando el diagnóstico de variantes de proceso en registros de eventos con ruido y ausencia de información.
- La validación realizada arrojó resultados favorables respecto a la calidad de los modelos descubiertos por la herramienta que implementa el algoritmo propuesto, ante registros de eventos con ruido y ausencia de información.

Recomendaciones

- Incorporar al algoritmo de competición otros patrones de control de flujo más complejos como la Selección no exclusiva.
- Implementar la validación de más restricciones para el reconocimiento de patrones de control de flujo.
- Realizar la validación con el cálculo de otras métricas de calidad y otros registros de eventos.

Bibliografía

A business process mining application for internal transaction fraud mitigation. **JANS, Mieke, VAN DER WERF, Jan Martijn, LYBAERT, Nadine y VANHOOF, Koen. 2011.** 10, septiembre 2011, Expert Systems with Applications, Vol. 38, pp. 13351-13359.

A multi-dimensional quality assesment of state-of-the-art process discovery algorithms using real-life event logs. **DE WEERDT, J., DE BACKER, M., VANTHIENEN, J. Y BAESENS, B. 2012.** 7, 2012, Information Systems, Vol. 37, pp. 654-676.

A Process Deviation Analysis Framework. **DEPAIRE Benoit, SWINNEN Jo, JANS Mieke y VANHOOF Koen. 2013.** Berlín : s.n., marzo 1, 2013, Springer, pp. 701-706. 978-3-642-36284-2.

AALST., W M P van der. 2011. *Process Mining. Discovery, Conformance and Enhancement of Bussiness Processes.* London : s.n., 2011. 978-3-642-19344-6.

Abstractions in Process Mining: A taxonomy of patterns. **BOSE R.P.J.C., AALST, W.M.P. van der.,. 2009.** Berlín : Springer, 2009, pp. 159-175.

An Experimental Evaluation of Passage-Based Process Discovery. **VERBEEK., H.M.W. y AALST., W.M.P. van der. 2012.** 2012, BPM Center Report.

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

Business Process Management. Concepts, Languages, Architectures. **WESKE M. 2007.** 2007. 978-3-540-73521-2.

Bussines Process Management(BPM): The Third Wave. **Fingar, H. Smith and P. 2003.** 2003, Meghan-Kiffer Press. 0929652339.

Canós, Jose H. 2008. *Metodologías Ágiles en el Desarrollo de Software.* 2008.

Conformance checking of processes based on monitoring real behavior. **ROZINAT, A. and AALST W.M.P. VAN DER. 2008.** 2008, Information Systems, pp. 64-65. 0306-4379.

Constructing Suffix Tree for Gigabyte Sequences with Megabyte Memory. **CHEUNG C.F., YU J.X., LU H.,. 2005.** 2005, IEEE TRans. Knowl. Data Eng.

Data Transformation and Semantic Log Purging for Process Mining. **LY, Linh Thao, INDIONO, Conrad, MANGLER, Jurgen y RINDERLE-MA, Stefanie. 2012.** junio 2012, Springer.

Diagnóstico de proceso basado en el descubrimiento de subprocesos. **YZQUIERDO-HERRERA., R., SILVERIO CASTRO, R., LAZO CORTÉS, M. Y TORRES GRAÑA, A. 2012.** 2, 2012, Ingeniería Industrial, Vol. 33.

Distributed Genetic Process Minig. **BRATOSIN., C., SIDOROVA., N. Y AALST W.M.P. Van der. 2010.** Barcelona, Spain : s.n., 2010. IEEE Worl Congress on Computational Intelligence (WCCI). pp. 1951-1958.

Flexible Heuristic Miner(FHM). **WEIJTERS. A.J.M.M., y RIBEIRO., J.T.S. 2011.** Paris : s.n., 2011, Computacional Intelligence and Data Mining (CIDM), pp. 310-317.

Fuzzy Mining – Adaptive Process Simplification Based on Multi-perspective Metrics. **GÜNTHER, Christian W., Aalst, Wil M.P. van der. 2007.** Brisbane, Australia : Springer, 2007. pp. 328-343.

Genetic Miner process minnig: An experimental evaluation. **DE MEDEIROS., A.K.A., WEITERS., A.J.M.M. Y AALST., W.M.P. van der. 2007.** 2, 2007, Data Mining and Knowledge Discovery, Vol. 14, pp. 245-304.

GÜNTHER C .W. 2009. *Process Mining in Flexible Enviroments.* University of Technology. Eindhoven : s.n., 2009.

GÜNTHER, C W. 2009. XES Standard Definition. [Online] 2009. www.xes-standard.org.

Llanes-Hernández, Alkaid Cruz, Pino-Marrero, Maikel. 2012. *Algoritmo para la detección de subprocesos mediante la alineación de las trazas usadas en la minería de procesos.* 2012. Tesis de Pregrado.

MA, L. 2012. *How to evaluate the Performance of Process Discovery Algorithms.* Eindhoven University of Technology. 2012. Master Thesis.

Mining Context-Dependent and Interactive Business Process Maps using Execution Patterns. **LI J., BOSE R.P.J.C.,. 2010.** s.l. : Springer, 2010, BPM 2010 Workshops, pp. 109-121.

Mining process models with non-free-choice constructs. **WEN, L., VAN DER AALST, W. M.P., WANG, J. Y SUN, J. 2007.** 2, 2007, Data Mining and Knowledge Discovery, Vol. 15, pp. 145-180.

PÉREZ ALFONSO, Damian. 2014. *Técnica para el diagnóstico de variantes de procesos de negocio.* La Habana : s.n., 2014.

Process diagnostics using trace aligment: Oportunities, issues, and challenges. **BOSE J. C. R. P., AALST W. M. P. van der.,. 2012.** 2, 2012, Information Systems, Vol. 37, pp. 117-141.

Process diagnostics using trace alignment: Opportunities, issues, and challenges. **BOSE R. P. J. C., AALST W. M. P. van der.,. 2012.** 2012, Information Systems, Vol. 37, pp. 117-141. 0306-4379.

Process discovery using integer linear programming. **VAN DER WERF., J., VAN DONGEN., B., HURKENS., C. y SEREBRENIK., A. 2008.** 2008, Applications and Theory of Petri Nets, pp. 368-387.

Process mining and petri net synthesis. **KINDER., E., RUBIN., V. Y SCHAFER., W. 2006.** Vienna : Springer, 2006, Business Process Management Workshops, pp. 105-116.

Process mining with the heuristics miner-algorithm. **WEIJTERS, A., VAN DER ALLST, W. M. P. Y DE MEDEIROS, A. K. A. 2006.** 2006, Vol. 166.

Process Mining: a two-step approach to balance between underfitting and overfitting. Software and Systems Modeling. **AALST. W.M.P. van der., Rubin, V., Verbeek, H.M.W., van Dongen, B.F., Kindler, E., Gunther, C.W. 2009.** 2009.

Process mining: discovering workflow models from event-based data. **WEIJTERS, A. y VAN DER AALST, W. M. P.** Proceedings of the 13th Belgium-Netherlands Conference on Artificial Intelligence. pp. 283–290.

Process Mining: Fuzzy Clustering and Performance Visualization. **Dongen, B.F. van, Adriansyah A. 2009.** 2009.

Process Mining: Overview and Outlook of Petri Net Discovery Algorithms. **VAN DONGEN, B., ALVES DE MEDEIROS, A. Y WEN, L. 2009.** 2009, Springer, pp. 225-242.

Process Trace Identification from Unstructured Execution Logs. **DESAI, N. BHAMIDIPATY, A., SHARMA, B., VARSHNEYA, V. K., VASA, M. Y NAGAR, S.,. 2010.** 2010. IEEE International Conference. pp. 17-24.

PUPO HERNÁNDEZ, EUDEL. 2014. *Algoritmo para el diagnóstico de variantes de proceso en registros de eventos con ruido, ausencia de información y alto número de sucesiones indirectas entre eventos.* La Habana : s.n., 2014.

Rediscovering Workflow Models fromn Event-Based Data using Litle Thumb. **WEIJTERS., A.J.M.M. Y AALST., W.M.P. van der. 2003.** 2, 2003, Integrated Computer-Aided Engineering, Vol. 10, pp. 151-162.

S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst. 2013. *Discovering Block-Structured Process Models from Event Logs A Constructive Approach.* 2013.

Shazia Sadiq, Pnina Soffer and Hagen Völzer. 2014. *Business Process Management Proceedings.* Haifa, Israel : Springer, 2014.

Sub-process discovery: Opportunities for Process Diagnostics. **YZQUIERDO-HERRERA., Raykenler., Silverio-Castro., Rogelio., Lazo-Cortés., Manuel. 2012.** GHENT : Springer, 2012. CONFENIS.

Supporting process mining by showing events at a glance. **SONG M., AALST W.M.P. van der.,. 2007.** 2007. Proceedings of the 17th Annual Workshop on Information Technologies and Systems(WITS). pp. 139-145.

The impact of enterprise systems on corporate performance: A study of ERP, SCM and CRM system implememntations. **HENDRICKS, K., SINGHAL, V. R. Y STRATMAN, J. K. 2007.** 1, septiembre 19, 2007, Journal of Operations Management, Vol. 25, pp. 65-82.

Towards Robust Conformance Checking. **ADRIANSYAH A., DONGEN, B.F., AALST, W.M.P. van der. 2011.** 2011, Preceedings of the 6th Workshop on Bussines Intelligence (BPI2010).

Trace Aligment in Process Mining: Oppotunities for Process Diagnostic. **BOSE R.P.J.C., AALST W.M.P. van der.,. 2012.** 2012.

Utilización de técnicas de minería de proceso en el entorno empresarial cubano. **PÉREZ ALFONSO, D., YZQUIERDO HERRERA, R., SILVERIO CASTRO, R. Y LAZO CORTÉS, M. 2012.** Villa Clara : s.n., 2012.

van der Aalst, W.M.P, Leemans, S.J.J and Fahland, D. 2013. *Discovering Block-Structured Process Models From Event Logs - A Constructive Approach.* 2013.

Workflow Mining: A Survey of Issues and Approaches. **AALST., W.M.P. van der, VAN DONGEN., B.F., J. HERBST, L., MARUSTER.**

Workflow Mining: Discovering process models from events logs. **AALST., W M P van der, WEIJTERS., A J M M y MARUSTER., L. 2004.** 9, 2004, IEEE Transactions on Knowledge and Data Engineering, Vol. 16, pp. 1128-1142.

Workflow Patterns. **AALST, W M P Van Der, HOFSTEDE, A H M Ter, KIEPUSZEWSKI, B y BARROS, A P. 2003.** 2003. Distrib. Parallel Databases. Vol. 14, pp. 5-51.

XES, XESame and Prom 6. **VERBEEK., H.M.W., BUIJS., J.C.A.M., DONGEN., B.F. Y AALST, W.M.P. van der. 2011.** 2011, Information Systems Evolution, pp. 60-75.

XES. XESame, and ProM 6. **VERBEEK, H. M. W., BUIJS, J. C. A. M., DONGEN, B. F. y AALST, W.M.P. 2011.** 2011, Information Systems Evolution.

YZQUIERDO HERRERA, Raykenler. 2012. *Modelo para la estimación de información ausente en las trazas usadas en la minería de proceso.* 2012.