Ministerio de Educación Superior

Universidad de la Ciencias Informáticas

Facultad 3

Centro para la Informatización de la Gestión de Entidades

Trabajo para optar por el grado de Doctor en Ciencias Técnicas

Modelo para la estimación de información ausente en las trazas usadas en la minería de proceso.

Autor: MSc. Raykenler Yzquierdo Herrera

Tutores: Dr. Rogelio Silverio Castro

Dr. Manuel Lazo Cortés

Ciudad de la Habana

Julio de 2012

DEDICATORIA

A mi familia. De manera muy especial a mi abuelo, a mi madre y a Nis.

A todos los que no se rinden nunca.

AGRADECIMIENTOS

Agradezco a todos los que de alguna forma han contribuido a mi formación. Especialmente agradezco:

A la Revolución.

A mis dos tutores, Dr. Rogelio Silverio Castro y Dr. Manuel Lazo Cortés, los cuales se han convertido en verdaderos amigos.

Al Dr. Ricardo Grau por su apoyo y asesoría.

A Yvonne Collada Peña por su apoyo y confianza.

A los muchachos Damián, Alkaid, Maikel y Yaidel por su colaboración y confianza.

A mis hermanos Rafael, Carlos, Alfredo, Ernesto y Davel los cuales me han apoyado incondicionalmente.

A la dirección del PEFCI por su apoyo y asesoría.

SÍNTESIS

La mayoría de las empresas utilizan sistemas de información para gestionar la ejecución de sus procesos de negocio. Estos registran en forma de trazas las acciones que se van realizando cuando se ejecutan instancias o casos del proceso de negocio. Al descubrimiento, monitoreo y mejora del proceso a partir de la información contenida en las trazas se le denomina minería de proceso. La mayoría de los algoritmos de minería de proceso parten del supuesto que las trazas son completas y están libres de ruido. En la realidad este supuesto pocas veces se cumple. La ausencia de información afecta la estructura y comprensión del modelo descubierto. En el presente trabajo se hace un análisis de las principales investigaciones que han abordado el tratamiento de la ausencia de información en las trazas usadas en la minería de proceso. Se describe y argumenta un nuevo modelo para la estimación de la información ausente en las trazas utilizadas en la minería de proceso. Como parte de la estimación se propone alinear las trazas durante la etapa de pre-procesamiento de estas para detectar las posibles situaciones en las que se manifiesta la ausencia de información. A partir de las situaciones detectadas se estima la información ausente y se genera en correspondencia un nuevo registro de evento, el cual puede ser utilizado por los diferentes algoritmos de descubrimiento de proceso existentes. Finalmente se discuten los resultados obtenidos a partir de la aplicación de la propuesta.

Palabras clave: ausencia de información, minería de proceso, registro de evento, traza

ÍNDICE

INTRODUCCIÓN

Actualidad del tema

En la actualidad las empresas líderes del mercado aplican estrategias de optimización de los procesos de negocio como la clave para reducir costos y expandirse en su entorno. Según un estudio reciente realizado por IBM, las empresas que mayor rendimiento están teniendo en sus sectores, son las que se han centrado en aumentar la agilidad para adaptarse a los cambios mediante buenas prácticas de trabajo y la correcta gestión de los procesos de negocio. El uso de sistemas de información que soportan la gestión de los procesos de negocio ha aumentado, lo que se evidencia en los ingresos que está obteniendo IBM en su línea de negocio enfocada a la gestión de procesos empresariales y software de integración [1].

Los Sistemas para la Gestión de Procesos de Negocio (BPMS) [2,3] posibilitan que se realice el modelado, la automatización y el análisis de los procesos de la empresa. Esto permite una mejor comprensión del negocio y un mejor desempeño en este [4,5]. Otros tipos de sistemas de información, tales como los sistemas de Planificación de Recursos Empresariales (ERP), sistemas de Gestión de Cadena de Suministros (SCM) y sistemas de Gestión de la Relaciones con los Clientes (CRM) posibilitan también gestionar los procesos de negocio de la organización [6-9]. Estos sistemas soportan las diferentes actividades que se realizan como parte de un proceso de negocio que puede estar o no explícitamente modelado. Los sistemas de información registran trazas de las acciones que se realizan durante la ejecución de las instancias o casos del proceso de negocio. Un registro de evento está constituido por un grupo de trazas. Las trazas se componen por una secuencia de eventos ordenados según su ocurrencia. Los eventos contenidos en una traza pueden tener asociados entre otros datos: el tiempo en el que se ejecutó, el

subsistema en el que se produjo y el usuario que ejecutó o inició la actividad. Las técnicas de minería de proceso permiten explicitar el conocimiento contenido en las trazas [10-12]. La idea de la minería de proceso es descubrir, monitorear y mejorar los procesos reales a través de la extracción del conocimiento contenido en los registros de evento ampliamente disponibles en los actuales sistemas de información [13,14].

La minería de proceso es un área de investigación relativamente nueva que permite hacer un análisis objetivo de la ejecución del proceso de negocio, es un puente importante entre la minería de datos y la modelación y análisis de procesos de negocio. Este tipo de técnicas se incluye en el área de la Inteligencia de Negocios (Business Intelligence, BI), área que ha tenido un auge importante en los últimos años [15].

El descubrimiento de un modelo de proceso representativo de la realidad es uno de los beneficios, pero no el único que brinda la minería de proceso [16-18]. También es posible realizar chequeos de conformidad para detectar desviaciones a partir de la comparación entre el registro de evento y el modelo representativo del proceso [19-21]. Es posible descubrir redes sociales y modelos organizacionales. Además, se puede extender el modelo de proceso considerando un análisis de su funcionamiento y la construcción automática de modelos de simulación. Las técnicas de minería de proceso permiten hacer recomendaciones y predicciones a partir del análisis de los datos actuales y de los históricos [13, 14, 22].

Un proceso puede ser analizado considerando las siguientes tres perspectivas o dimensiones: la perspectiva de control de flujo, la de los recursos (también llamada organizacional) y la de casos. En dependencia de la información contenida en los registros de eventos y la problemática a resolver se seleccionan las perspectivas a analizar. La más abordada de las perspectivas es la de control de flujo, en la cual es

posible determinar la estructuración del flujo de actividades del proceso de negocio. En esta dimensión se debe contar con información relativa a la identificación del caso o instancia ejecutada, la identificación de la actividad y el tiempo en el que se produjo la actividad. En la perspectiva de recurso es posible determinar la estructura organizacional con respecto a la cual se realizan las diferentes actividades, esto permite conocer la jerarquía que se establece entre roles y usuarios durante el desarrollo del proceso de negocio. Se debe contar con información referente al posible usuario del sistema que ejecutó cada actividad. En la tercera perspectiva relacionada con los casos se puede descubrir casos excepcionales, como la probabilidad de que, personas de determinado sexo realicen en tiempo determinada actividad. Se debe contar con información detallada de las actividades, como puede ser, la edad y el sexo del usuario involucrado [23, 24].

Las diferentes técnicas de descubrimiento desarrolladas buscan identificar a partir del registro de evento un modelo representativo del proceso en el que queden reflejadas las dependencias entre las diferentes actividades que han sido registradas. Los algoritmos existentes manejan generalmente un conjunto de constructores de flujo de trabajo que permiten reconocer una determinada estructura o patrón que aparece de manera común en el proceso [25]. Estos constructores de flujo de trabajo son: secuencia, paralelismo, selección, lazos, no libre selección (non-free-choice), tareas invisibles y tareas duplicadas. Aun cuando existen trabajos que cubren la totalidad de los constructores de flujo de trabajo enunciados, la mayoría de los algoritmos desarrollados dan un cubrimiento parcial a estos [16, 26]. Los algoritmos de minería de proceso existentes pueden tener problemas para manejar determinados constructores de flujo de trabajo debido a que la notación que utilizan para representar el modelo de proceso descubierto

no los soporta [13, 23, 27], además, en situaciones reales la minería de proceso se enfrenta al ruido presente en el registro de evento y la desestructuración del proceso analizado [13, 23, 28-30].

Formulación del problema

La mayoría de los algoritmos de minería de proceso parten del supuesto que las trazas son completas y están libres de ruido. En la realidad este supuesto rara vez se cumple [13, 31, 32]. La completitud puede verse en dos sentidos. La primera manifestación se refleja en que determinadas instancias de un proceso pueden no estar registradas en las trazas debido a que no han ocurrido, aun cuando dichas instancias son soportadas por los sistemas de información empleados en la empresa. Esta situación puede afectar el modelo descubierto [13, 16] por lo cual, trabajos como [33] se han dirigido en la solución de esta problemática. La segunda manifestación se refleja en que una o varias actividades del proceso no se registran en las trazas. A esto se le denominará ausencia de información. Una actividad no queda reflejada en el registro de evento si no fue informatizada, si fue informatizada pero el sistema de información no deja constancia de su ocurrencia en el registro de evento o si el registro de evento fue afectado y presenta ruido [34]. A este tipo de actividad que no queda reflejada en el registro de evento se le denomina actividad o tarea invisible [16, 23, 35].

Los sistemas de información usados para informatizar un determinado proceso de negocio cubren un subconjunto de actividades, es decir, durante la ejecución del proceso de negocio pueden realizarse de manera intercalada tanto actividades de las cuales queda evidencia en el registro de evento como actividades de las cuales no queda evidencia. En consecuencia, los sistemas de información registran en las trazas una secuencia incompleta de las actividades que conforman las instancias del proceso de

negocio. Esta investigación se enfoca en esta manifestación de la ausencia de información.

La ausencia de información puede generar que los algoritmos dirigidos a descubrir el proceso de negocio ejecutado obtengan modelos en los que se reflejan incorrectas relaciones entre las actividades, siendo este un factor determinante en la falta de estructuración de los modelos descubiertos. Es necesario señalar que no siempre la ausencia de información produce afectaciones en el modelo descubierto, pero por lo general, la ausencia de información dificulta la comprensión del modelo descubierto [36, 37].

El modelo descubierto se apoya en un grupo de patrones de flujo de trabajo para cubrir completamente el comportamiento reflejado en las trazas, sin embargo, ante situaciones de ausencia de información los modelos obtenidos por los diferentes algoritmos de descubrimiento de procesos pueden diferenciarse significativamente, es decir, existe ambigüedad en la interpretación de las trazas [23]. Para una correcta comprensión del modelo descubierto, es útil reflejar explícitamente, de ser posible, las actividades invisibles. Algunas técnicas de descubrimiento reflejan en los modelos obtenidos las actividades invisibles [38-40].

Se debe señalar que cuando estamos frente a las trazas generadas por BPMS es poco frecuente la ausencia de información debido a que las instancias del proceso se ejecutaron a partir de un modelo de proceso explícito en la herramienta utilizada. Sin embargo, actividades como “Realizar llamada telefónica” pueden no quedar registradas.

Antes de aplicar algún algoritmo de descubrimiento se tratan algunas situaciones que pueden estar asociadas a manifestaciones de ruido, esto se realiza durante la etapa de pre-procesamiento. En este sentido, se realizan acciones de verificación, limpieza,

agrupamiento y alineado de las trazas, lo cual no significa que se detecten y corrijan todas las manifestaciones de ausencia de información [13, 29, 30, 32, 41-44]. Otras investigaciones se han centrado en el desarrollo de algoritmos que son robustos ante el ruido y ante algunas situaciones en las que existe ausencia de información [12, 23, 39, 45-48].

La ausencia de información ha sido abordada directamente desde la perspectiva del constructor de actividades invisibles o variables escondidas. Dada su importancia este aspecto ha sido abordado en recientes evaluaciones realizadas a los algoritmos de descubrimiento más efectivos [27]. Es necesario resaltar que, existen situaciones complejas en las que existe ausencia de información y no pueden ser manejadas correctamente por los algoritmos desarrollados hasta el momento [16, 23, 49, 50]. En consecuencia se formaliza el siguiente problema científico:

- ¿Cómo disminuir la afectación que provoca la ausencia de información en las trazas sobre la estructura y comprensión del modelo de proceso descubierto a partir de la aplicación de técnicas de minería de proceso?

Hipótesis

Un modelo para la estimación de la información ausente en las trazas utilizadas en las técnicas de minería de proceso, basado en su alineación, permite descubrir modelos de procesos más estructurados y comprensibles.

Objetivo general

Desarrollar un modelo para la estimación de información ausente en las trazas utilizadas por las técnicas de minería de proceso, basado en su alineación, para el descubrimiento de modelos de proceso más estructurados y comprensibles.

Para lograr este objetivo se plantean los siguientes **objetivos específicos**:

1. Desarrollar un algoritmo general para la descomposición del proceso analizado a partir de la alineación de las trazas, considerando los patrones básicos de flujo de trabajo.
2. Desarrollar un algoritmo para la estimación de la información ausente en las trazas de un proceso a partir de su descomposición.
3. Desarrollar una herramienta informática basada en los algoritmos propuestos y que facilite la validación del modelo desarrollado para la estimación de información ausente en las trazas usadas en la minería de proceso.
4. Validar el modelo desarrollado a partir de su aplicación.

Métodos y técnicas a utilizar

Entre las estrategias de investigación que se utilizan están la exploratoria y la explicativa.

Se exploraron diferentes técnicas y tendencias en el descubrimiento de procesos basados en técnicas de minería de proceso, así como situaciones en las que se manifiesta ausencia de información con vistas a desarrollar un modelo computacional novedoso y que resuelva las deficiencias reportadas en la literatura.

Métodos teóricos: histórico lógico, hipotético deductivo, analítico sintético y sistémico.

Se enfocaron las problemáticas asociadas a la ausencia de información en las trazas usadas en técnicas de minería de proceso desde un enfoque histórico lógico, en la primera parte de la investigación se desarrolló un estudio del estado del arte de la problemática analizada; detallando las bondades y deficiencias de cada uno de los métodos y las tendencias en la resolución de esta problemática.

La investigación sigue además un método hipotético deductivo porque a partir del problema concreto se plantean objetivos específicos e hipótesis que en el transcurso de la investigación son resueltas siguiendo métodos científicamente bien fundamentados.

El método analítico sintético se sigue al descubrir los distintos elementos que componen la naturaleza o esencia asociada al fenómeno de ausencia de información en las trazas. Definiéndose las causas y los efectos, para posteriormente integrar los elementos en una unidad nueva, en una comprensión total de la esencia de lo que ya se conoce en todos sus elementos y particularidades.

En cada caso se plateó el problema como un todo, donde las trazas utilizadas, la propia dinámica de aplicación de técnicas de minería de proceso en el descubrimiento de procesos y las técnicas computacionales desarrolladas para la estimación de información ausente en las trazas se funden en un sistema sostenible e integral.

Método empírico: experimentación y medición.

Además de seguir métodos teóricos se siguen los métodos empíricos basando la investigación en la experimentación con datos provenientes de situaciones reales suministrados por sistemas utilizados durante el proceso de desarrollo de software.

Se aplicaron pruebas estadísticas bien fundamentadas para analizar la efectividad del modelo desarrollado y la calidad de las respuestas finales. Se establecieron estadígrafos e indicadores adecuados que permiten realizar correctas mediciones de los resultados.

Los principales aportes de esta investigación se reflejan a continuación:

Aporte teórico

- La concepción y fundamentación de un modelo para la estimación de información ausente en trazas que contribuye a la mejora de la estructura y

comprensión de los modelos de procesos descubiertos usando técnicas de minería de proceso.

- Dos nuevas manifestaciones de ausencia de información que no había sido reportado en la literatura.

Aporte práctico

- Un algoritmo general para la descomposición del proceso analizado a partir de la alineación de las trazas, considerando los patrones básicos de flujo de trabajo.
- Fundamentación de los patrones para la identificación de la ausencia de información a partir de la alineación de las trazas.
- Herramienta para la determinación de la ausencia de información en las trazas que posibilita que las técnicas de descubrimiento de proceso obtengan modelos más estructurados y comprensibles.

Estructura del trabajo

El presente trabajo está conformado por introducción, tres capítulos, las conclusiones y las recomendaciones.

Capítulo 1: Se discute las ideas básicas y las diferentes estrategias utilizadas en el manejo de ausencia de información en el área de la minería de proceso. Se hace una evaluación crítica de las ventajas y desventajas de los diferentes enfoques.

Capítulo 2: Se describe y fundamenta un modelo para la estimación de información ausente en trazas generadas por los sistemas de información y usadas en las técnicas de minería de proceso para descubrir el proceso ejecutado.

Capítulo 3: Se presenta la implementación del modelo propuesto y su aplicación. Se muestra la validación de la solución propuesta.

CAPÍTULO 1. FUNDAMENTOS TEÓRICOS Y METODOLÓGICOS

En este capítulo se aborda un conjunto de aspectos relacionados con la minería de proceso. Se hace una revisión de las diferentes aristas desde las que ha sido analizado el problema de ausencia de información en las trazas, considerando los trabajos de los autores más referenciados en el área de minería de proceso. Además se valoran un conjunto de técnicas que pueden ser de utilidad para la estimación de información ausente en las trazas.

1.1 Minería de proceso

Las técnicas de minería de proceso, permiten extraer información no trivial y útil de los registros de evento almacenados por sistemas de información. Un elemento fundamental en la minería de proceso es el descubrimiento del control de flujo, es decir, la construcción automática del modelo de proceso en el que se describen las dependencias causales entre las actividades del proceso [13].

Un concepto necesario para entender la minería de proceso es el de *proceso de negocio*, este ha sido definido por varios autores. Algunas de las definiciones estudiadas se enuncian a continuación.

La Workflow Management Coalition (1999) [51] define como proceso de negocio: "Un conjunto de uno o más procedimientos vinculados o actividades que en conjunto tributan a un objetivo de negocio o un objetivo de política, por lo general en el contexto de una estructura organizacional que define roles funcionales y sus relaciones".

Según Smith y Fingar (2003) [52] "Un proceso de negocio es el conjunto completo y coordinado de actividades colaborativas y transaccionales que proporcionan valor a los clientes".

Según Mathias Weske (2007) [5] "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".

El autor del presente trabajo asume como la definición de proceso de negocio la dada por Mathias Weske (2007). Esta definición permite entender claramente las posteriores valoraciones realizadas sobre este concepto.

La minería de proceso provee un importante puente entre la minería de datos, el modelado de los procesos de negocio y análisis de estos. Bajo el área de BI (siglas de Business Intelligence) [53-55] se han difundido un grupo de términos que encierran diferentes tipos de análisis en este contexto, tales como, BAM (siglas de Business Activity Monitoring) referido a las tecnologías que hacen posible un análisis en tiempo real de los procesos de negocio [56, 57], CEP (siglas de Complex Event Processing) referido a las tecnologías que permiten procesar grandes cantidades de eventos para monitorear, guiar y optimizar el negocio en tiempo real [58, 59], CPM (siglas de Corporate Performance Management) referido a la medición del funcionamiento del proceso o la organización [60, 61]. Otros términos están vinculados con la gestión, como es el caso de CPI (siglas de Continuous Process Improvement) [62, 63], BPI (siglas de Business Process Improvement) [64, 65], TQM (siglas de Total Quality Management) [66, 67] y Six Sigma [68]. Las investigaciones en estas áreas tienen en común que los procesos son "empujados bajo el microscopio" con el objetivo de identificar posibles mejoras. La minería de proceso puede considerarse una tecnología que contribuye a cada una de las áreas antes mencionadas [13, 14, 28, 69].

En la última década se ha hecho común el tratamiento del registro de evento y cada vez más sistemas de información incorporan técnicas de minería de proceso. Algunos de estos software son: ARIS Process Performance Manager (desarrollado por Software AG) [70], Comprehend (desarrollado por Open Connect) [71], Discovery Analyst (desarrollado por StereoLOGIC) [72], Flow (desarrollado por Fourspark) [73], Futura Reflect (desarrollado por Futura Process Intelligence) [74], Interstage Automated Process Discovery (desarrollado por Fujitsu) [75], OKT Process Mining suite (desarrollado por Exeura) [76], Process Discovery Focus (desarrollado por Iontas/Verint) [77], ProcessAnalyzer (desarrollado por QPR) [78], ProM (desarrollado por TU/e) [79], Rbminer/Dbminer (desarrollado por UPC) [80], y Reflect | one (desarrollado por Pallas Athena) [81].

Como punto de partida para la aplicación de técnicas de minería de proceso están las trazas, las cuales han sido generadas por uno o varios sistemas de información usados en una empresa. Varias han sido las definiciones dadas para registro de evento y traza [23, 24, 39, 82, 83]. Por lo general estas definiciones coinciden y se resumen en la dada por Van der Aalst [84].

Definición 1.1 (Traza y registro de evento). Se denota por Σ el conjunto de todas las actividades. Σ^+ es el conjunto de todas las secuencias finitas de actividades no vacías sobre Σ. Cada $T \in \Sigma^+$ es una posible traza. Un registro de evento $\mathcal{L}$ es un grupo de trazas.■

Para la representación de un registro de evento se han definido dos estándares Mining eXtensible Markup Language (MXML) y eXtensible Event Stream (XES) [26, 85-97].

Un registro de evento constituye la entrada de los tres tipos de técnicas de minería de proceso. Estos tipos de técnicas se muestran en la Figura 1.1.

Figura 1.1. Representación de los tres tipos fundamentales de técnicas de minería de proceso: descubrimiento, conformidad y mejoramiento (tomado de [14]).

El primer tipo de técnica de minería de proceso es el descubrimiento. Este tipo de técnica permite descubrir un modelo representativo del proceso ejecutado en la empresa a partir de un registro de evento. Existen una gran variedad de investigaciones desarrolladas en este sentido, entre los algoritmos desarrollados se puede mencionar: Alpha [18, 88-91], Genetic Miner [23], Heuristic Miner [47, 92], Transition System Miner [17], Fuzzy Miner [48], entre otros. Los modelos descubiertos a partir de la aplicación de técnicas de minería de proceso pueden ser representados utilizando diferentes notaciones, entre las más empleadas se encuentran las Redes de Workflow [26, 93, 94], Event-driven Process Chains (EPCs) [95], Transition System (TS) [96], Business Process Modeling Notation (BPMN), diagrama de actividades Unified Modeling Language (UML) [14].

El segundo tipo de técnica está asociado con la conformidad o con el chequeo de conformidad [20, 36, 37, 97-99]. La idea fundamental de estas técnicas es comparar un modelo de proceso existente con un registro de evento del mismo proceso. El chequeo de conformidad es un análisis que permite saber hasta qué punto el registro de evento se

corresponde con el modelo de proceso y viceversa. Diferentes tipos de modelos pueden ser usados en el chequeo de conformidad, tales como, modelos normativos o descriptivos, modelos organizacionales, reglas y políticas del negocio, leyes, etc.

Por último se encuentra la mejora. Este tipo de técnica permite la extensión del conocimiento que se tiene del proceso de negocio o la mejora de este. A partir de un modelo de proceso existente y el registro de evento correspondiente al mismo proceso se detectan aspectos como: cuellos de botella, niveles de servicio, tiempos de espera y ejecución, frecuencia de algún evento, entre otros. Estos aspectos pueden reflejarse en un nuevo modelo de proceso [13, 100-102].

Como se mencionó anteriormente un proceso puede ser analizado considerando las siguientes tres perspectivas, la perspectiva del control de flujo, la de los recursos y la de casos. Esto ha permitido que las técnicas de minería de proceso jueguen un papel importante en el ciclo de vida de la Gestión de Proceso de Negocio (en inglés Business Process Management, BPM). La Figura 1.2 muestra un esquema del ciclo de vida de BPM.

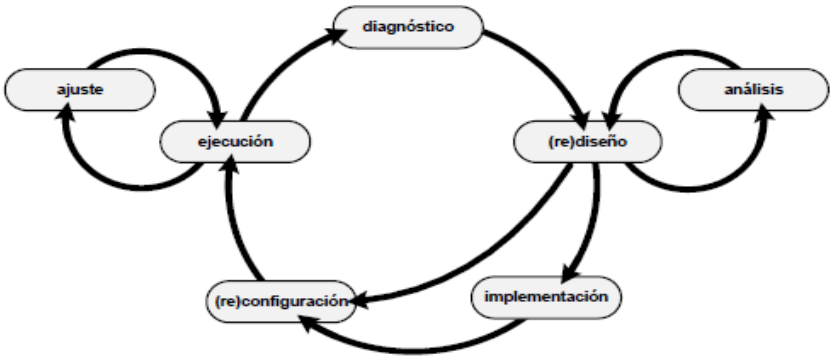

Figura 1.2. Ciclo de vida de BPM (tomado de [14]).

El ciclo de vida de BPM muestra las siete fases de un proceso de negocio y sus correspondientes sistemas de información. En la fase de (re)diseño un nuevo modelo de proceso es creado o uno existente es adaptado. En la fase de análisis el modelo

candidato y sus alternativas son analizados. Después del (re)diseño el modelo es implementado o en caso de existir una implementación puede ser (re)configurado considerando las modificaciones realizadas en el (re)diseño. En la etapa de ejecución el proceso informatizado se ejecuta. Durante la ejecución del proceso este es monitoreado, esto permite realizar pequeños ajustes (fase de ajustes) sin necesidad de realizar un rediseño del proceso. En la fase de diagnóstico el proceso ejecutado es analizado y los resultados pueden ser empleados en la fase de (re)diseño. Aunque la minería de proceso es utilizada fundamentalmente en la fase de diagnóstico puede utilizarse en otras fases. Por ejemplo en la fase de ejecución puede ser utilizada para el soporte operacional. En este sentido se pueden hacer recomendaciones y predicciones a partir de los modelos descubiertos usando los datos históricos y los datos actuales de la ejecución. En la etapa de (re)configuración pueden emplearse también los análisis realizados usando la minería de proceso [13, 14].

1.2 Ausencia de información

Descubrir un modelo de proceso requiere interpretar el comportamiento reflejado en las trazas y en correspondencia, reflejar en dicho modelo un grupo de patrones de flujo de trabajo [25, 103, 104]. No todos los algoritmos interpretan de la misma forma las trazas y manejan todos los patrones de flujo de trabajo [13, 23, 105].

Para que un modelo descubierto a partir del uso de técnica de minería de proceso sea útil es necesario que las trazas analizadas reflejen las instancias del proceso de la manera más exacta posible, de lo contrario se estaría infiriendo un comportamiento que no es representativo de la realidad.

En el presente trabajo se abordará la ausencia de información como la ausencia en las trazas de una o varias actividades ejecutadas como parte de las instancias del proceso,

debido a que estas no pueden ser registradas por los sistemas informáticos usados. A este tipo de actividad se le denominará *actividad invisible*.

Las actividades invisibles han sido abordadas en investigaciones asociadas al descubrimiento del modelo de proceso [13, 16, 23] o asociadas al chequeo de conformidad [20, 36, 37]. Entre los autores consultados existe un consenso en la definición de una actividad invisible.

A continuación se presentan las situaciones de ausencia de información descritas en la literatura [23], así como la interpretación que realizan los algoritmos de descubrimiento de las trazas afectadas por ellas. Esta ejemplificación se realiza utilizando Redes de Workflow [102]. Las actividades invisibles contenidas en el modelo de proceso se representan con recuadros grises.

Situación de salto. Una tarea invisible se puede manifestar cuando se produce un salto de una o varias actividades en una situación de selección. La Figura 1.3 muestra un ejemplo de esta situación y refleja el comportamiento registrado en la secuencia de actividades ABD, ACD, AD.

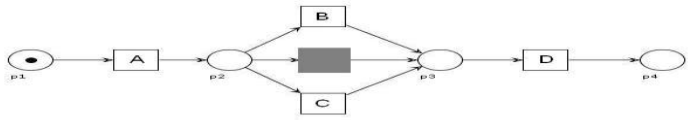

Figura 1.3. Red de Workflow, muestra una situación de salto.

Situación de división/unión. Una actividad invisible puede manifestarse en una situación en la que es necesario dividir la ejecución o unirla producto de un punto de selección, después de esta selección aparece una situación de paralelismo entre las actividades. La Figura 1.4 muestra un ejemplo de esta situación y refleja el comportamiento registrado en la secuencia de actividades ADE, ACBE, ABCE. En este caso, primero se produjo una selección y luego un paralelismo, pero otros casos se

pueden manifestar si el evento de inicio y el de fin de un subproceso no se reflejan en las trazas. En la Figura 1.4, el evento de inicio y el de fin están representados por las actividades invisibles, tomadas de izquierda a derecha.

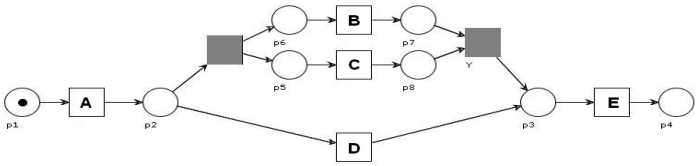

Figura 1.4. Red de Workflow, muestra una situación de división/unión.

La capacidad de detección por un algoritmo de descubrimiento de estos patrones está determinada en ocasiones, no solo por el algoritmo de descubrimiento, sino también por las posibilidades que tiene la notación utilizada para representar el modelo descubierto y específicamente las actividades invisibles [106]. Es necesario señalar que en los algoritmos analizados el tratamiento de las actividades invisibles no se realiza de manera explícita en todos los casos, es decir, al detectar una posible actividad invisible no se adiciona al modelo una nueva actividad. En ocasiones se obtiene un modelo que implícitamente puede reflejar la ausencia de información [23].

Algoritmos como el Alpha [18] no manejan el constructor de tareas invisibles debido a que asumen como precondición que las trazas son completas y están libres de ruido. Esto provoca que de aplicarse dichos algoritmos ignorando las precondiciones mencionadas, el modelo descubierto represente incorrectamente la realidad, aun cuando se cubra completamente el comportamiento descrito en las trazas.

Los siguientes ejemplos ilustran cómo la ambigüedad en las interpretaciones de las trazas puede manifestarse en situaciones en las que existe ausencia de información.

Actividades invisibles contra actividades duplicadas. Una misma actividad puede aparecer más de una vez en una misma traza, la mayoría de los algoritmos desarrollados consideran que cada actividad aparece una sola vez en cada traza, por lo cual ignoran las

actividades duplicadas o las manejan a partir del constructor de lazos (Figura 1.6 b)). Esta situación puede interpretarse también como ausencia de información.

La siguiente secuencia de actividades ABBC, puede reflejarse por los dos modelos representados en la Figura 1.5. En la Figura 1.5 a) se emplea el constructor de actividades duplicadas mientras que en b) se usa el constructor de actividades invisibles.

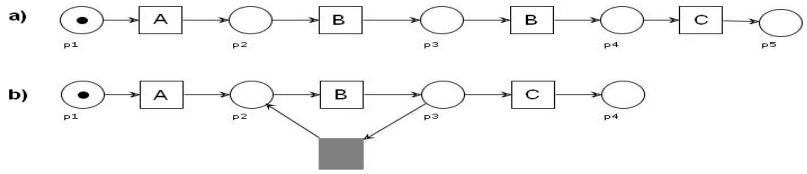

Figura 1.5. Red de Workflow, muestra en a) el uso del constructor de actividades duplicadas y en b) el constructor de actividades invisibles.

Actividades invisibles contra lazos. Esta es una situación que se puede considerar como una unión de la situación de *Actividades invisibles contra actividades duplicadas* con la *Situación de Salto*. Las siguientes secuencias de actividades AC, ABC, ABBC pueden reflejarse por los dos modelos representados en la Figura 1.6. En la Figura 1.6 a) se emplea el constructor de actividades invisibles y en b) se usa el constructor de lazos.

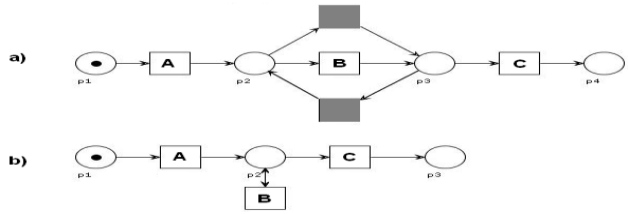

Figura 1.6. Red de Workflow, muestra en a) el uso del constructor de actividades invisibles y en b) el constructor de lazos.

Actividades invisibles contra sincronización. Esta situación puede considerarse como una generalización de la *Situación de división/unión*. Los dos modelos de la Figura 1.7 pueden reflejar las mismas secuencias de actividades (ABDG, ADBG, ACEFG, ACFEG) reflejadas en las trazas. En la Figura 1.7 en a) se emplea el constructor de XOR-split/join mientras que en b) se usa el constructor de actividades invisibles.

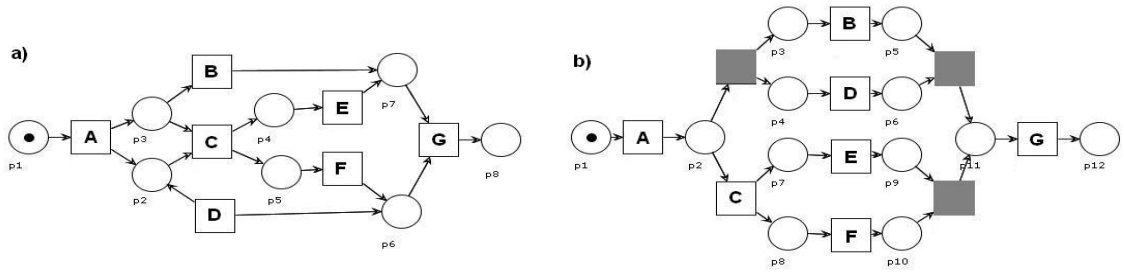

Figura 1.7. Red de Workflow, muestra en a) el uso del constructor de XOR-split/join y en b) el constructor de actividades invisibles.

Lazos contra actividades invisibles junto a actividades duplicadas. Esta situación se puede interpretar como una unión de la *Situación de Salto* con la *Situación de división/unión*. Los dos modelos de la Figura 1.8 pueden representar las secuencias de actividades ABCD, ACBD, AD, ABD, ACD. En la Figura 1.8 en a) se emplea el constructor de actividades invisibles junto al de actividades duplicadas mientras que en b) se usa el constructor de lazos. El modelo descubierto usando el constructor de lazos (Figura 1.8 b)) es más expresivo que el modelo de la Figura 1.8 a), es decir, sobrepasa el comportamiento reflejado en la secuencia de actividades expuestas.

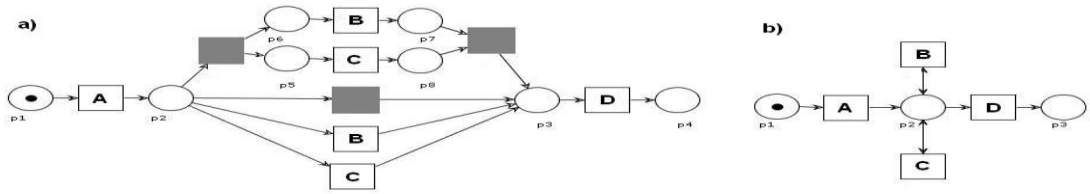

Figura 1.8. Red de Workflow, muestra en a) el uso del constructor de actividades invisibles unido al de actividades duplicadas y en b) el constructor de lazos.

Es necesario resaltar que algunos de los algoritmos desarrollados hasta el momento no manejan el constructor de actividades invisibles para la totalidad de las situaciones a las que se ha hecho referencia.

1.3 La ausencia de información y el descubrimiento del proceso

La ausencia de información en la minería de proceso ha sido tratada desde dos aristas fundamentales. Una está orientada a que las técnicas de descubrimiento sean robustas

ante situaciones de ausencia de información, mientras que la segunda arista está dirigida a tratar con este tipo de problema en la etapa de pre-procesamiento del registro de evento, lo cual es consecuencia de las deficiencias presentadas por las técnicas de descubrimiento en el tratamiento de situaciones asociadas fundamentalmente al ruido.

El término ruido ha sido empleado en varias ocasiones en esta investigación por lo cual es necesario explicar qué se entiende por este término. Varios han sido los trabajos que abordan el ruido en el registro de evento, autores como Folino [34] consideran que el ruido se refiere a situaciones donde el registro de evento es incompleto, contiene errores o refleja un comportamiento excepcional. Van der Aalst [13] se refiere al ruido como ese comportamiento reflejado en las trazas y que rara vez ocurre, que es excepcional o poco frecuente, es decir, que no se corresponde con el comportamiento típico observado en el proceso. Este autor no hace referencia a los errores porque ningún registro de evento revela explícitamente los errores que contiene. Por otra parte, se supone que los valores extremos corresponden a ese comportamiento excepcional más que a errores.

Para esta investigación se considera lo plantado por Van der Aalst respecto al ruido, dado que sin importar las causas (errores, completitud del registro de evento, acciones poco frecuentes) el ruido es ese comportamiento poco frecuente reflejado en las trazas.

En esta sección se hace un análisis de los trabajos relacionados con el descubrimiento de procesos mediante técnicas de minería de proceso y el tratamiento que se ha dado en estos a las tareas invisibles. Algunas de las investigaciones analizadas se implementaron como algoritmos que forman parte de la herramienta ProM [79, 107], dichos algoritmos se sometieron a un grupo de pruebas asociadas a cada una de las manifestaciones de ausencia de información antes expuestas.

Es necesario antes de analizar las técnicas de descubrimiento desarrolladas en la minería de proceso conocer en qué consiste el problema asociado al descubrimiento de un modelo de proceso a partir de un registro de evento. Varios han sido los autores que de una forma u otra han hecho referencia a este problema [11, 23, 33, 39, 108, 109]. Para su formalización el autor de este trabajo considera la definición dada por Van der Aalst (2011) [13] debido a que sintetiza lo abordado en los trabajos relacionados.

Definición 1.2 (Problema de descubrimiento de proceso): Sea $\mathcal{L}$ un registro de evento según la *Definición 1.1* o la especificación del estándar XES [86]. Un algoritmo de descubrimiento de proceso es una función que mapea $\mathcal{L}$ en un modelo de proceso tal que el modelo es "representativo" del comportamiento reflejado en el registro de evento. El reto es encontrar este tipo de algoritmo. ■

El término representativo se señala entre comillas debido a que el problema en el descubrimiento de proceso está básicamente en determinar un modelo que represente correctamente a la realidad reflejada en el registro de evento. La evaluación de la representatividad de un modelo puede hacerse desde varias aristas por lo cual no se ha identificado en las investigaciones analizadas un criterio formal en este sentido.

A continuación se analizan un conjunto de trabajos relacionados con el descubrimiento de procesos.

Cook y otros. Los primeros trabajos en la minería de proceso fueron realizados por estos autores en el área del desarrollo de software. Un análisis de los algoritmos desarrollados demuestra que los modelos descubiertos no eran del todo correctos y completos, sin embargo, se lograba identificar los principales patrones de control de flujo presentes en las trazas. Para la representación de los modelos se empleaban Máquinas de Estado Finito.

Cook participó en el desarrollo de tres algoritmos RNet, KTail y Markov, siendo este último el que mejor reflejaba las trazas analizadas. Markov es un algoritmo que tiene una base estadística y permite identificar los predecesores y sucesores de una actividad, para ello se apoya en una tabla de frecuencias en la que se refleja la probabilidad de que un evento ocurriese. Este algoritmo fue extendido posteriormente para manejar procesos concurrentes. Se agregaron cuatro métricas que permiten identificar puntos en los que aparecen relaciones XOR/AND-split/join. Este algoritmo es robusto ante el ruido debido a su basamento probabilístico [10, 11, 108, 110, 111].

La implementación de los algoritmos desarrollados se realizó en la herramienta DaGama, la cual forma parte del marco de trabajo Balboa [110].

En los algoritmos analizados no se hace un tratamiento de las tareas invisibles de manera explícita. Las investigaciones realizadas por estos autores permiten manejar las tareas invisibles en los casos de *Situación de salto* y *Situación de división/unión* [23].

Herbst y otros. Uno de los aspectos más importantes del trabajo de estos autores es que sus algoritmos manejaban las tareas duplicadas. Herbst participó en el desarrollo de tres algoritmos: MergeSeq, SplitSeq y SplitPar. Los algoritmos ejecutan dos pasos, el primero permite capturar las dependencias entre las tareas apoyándose en el uso de la métrica LLH (siglas de log-likelihood) desarrollada. Dicha métrica indicaba cuán bien el modelo expresado en SAG (siglas de Stochastic Activity Graph) expresaba el comportamiento recogido en las trazas. El segundo paso permite transformar el SAG en ADL (siglas de Adonis Defnition Language). El ADL es un lenguaje de estructuras en bloques que permite especificar modelos de flujo de trabajo. La conversión de SAG a ADL permite la creación de un modelo bien definido.

MergeSeq y SplitSeq son útiles para manejar procesos secuenciales, mientras que SplitPar puede manejar también procesos concurrentes. La implementación de los algoritmos desarrollados se realizó en la herramienta InWoLve [112-115].

Los algoritmos desarrollados pueden comportarse robustamente ante el ruido y manejan los constructores principales: secuencia, selección, paralelismo, lazos y tareas duplicadas. Sin embargo, no se hace un tratamiento de las actividades invisibles de manera explícita. Se manejan las actividades invisibles en la *Situación de salto* y la *Situación de división/unión* [23].

Schimm. Lo más significativo de sus trabajos es que el modelo de proceso descubierto es completo y mínimo. El modelo descubierto no generaliza un comportamiento más allá de lo observado en las trazas. Normalmente se detecta una relación de paralelismo entre dos tareas cuando estas aparecen intercambiadas en el orden de aparición en las trazas, sin embargo, Schimm identifica dos posibilidades para esta situación. Solo si el tiempo de inicio y fin de las tareas involucradas coinciden se identifica una situación de paralelismo. Schimm analiza el proceso mediante un grupo de reglas de rescritura que aplica sobre el álgebra de flujo de trabajo. Utiliza un modelo de estructura en bloque que permite formar correctos modelos. El autor realizó extensiones al álgebra de flujo de trabajo para poder representar en los modelos una sincronización parcial. Se concibe una fase de pre-procesamiento en la que se detectan aspectos como las tareas duplicadas y se trata el ruido. El algoritmo consta de seis pasos y puede manejar los principales operadores de control de flujo [26, 45, 46, 116, 117].

La implementación del algoritmo desarrollado se realizó en la herramienta Process Miner [118].

No se hace un tratamiento de las actividades invisibles de manera explícita y se manejan las tareas invisibles solo en la *Situación de salto* [23].

Grecco y otros. El aspecto que más resalta en los trabajos de estos autores es que se obtiene como resultado de la minería de proceso un árbol jerárquico que representa los diferentes niveles de abstracción del modelo de proceso. La raíz del árbol representa el modelo más general que encierra los aspectos comunes en las instancias del proceso reflejadas en las trazas. Los nodos que se encuentran entre la raíz y las hojas representan características comunes presentes en los nodos hijos. El algoritmo se divide en dos pasos, el primero consiste en obtener un modelo que cubra completamente las trazas analizadas, a partir de este modelo base se hace una selección de atributos para agrupar las instancias del proceso y formar particiones. Este proceso se realiza por cada nivel hasta que se cumplen las condiciones de parada. El segundo de los pasos permite construir las abstracciones correspondientes a cada nivel, para ello agrupa en un nodo las tareas comunes en sus hijos y agrupa en subprocesos las tareas particulares. Se emplea una red heurística para representar el modelo de proceso descubierto. El algoritmo no es robusto ante el ruido [119, 120].

El primero de los pasos fue implementado en el plugin Disjunctive Workow Schema (DWS) mining e incorporado a la herramienta ProM [79, 107].

Las pruebas realizadas tanto a este algoritmo, usando ProM, como a los que se mencionan a continuación arrojaron que, no se hace un tratamiento de las tareas invisibles de manera explícita y no se toma en cuenta evidencia que puede indicar una posible ausencia de información para enfrentar situaciones en las que existe ambigüedad en la interpretación de las trazas. Para cada uno de los algoritmos analizados, se muestran en la Tabla 1.2 los casos en los que el modelo resultante se distanció más de

una correcta interpretación. Entiéndase como una correcta interpretación los casos en los que el modelo resultante, aun cuando no refleja explícitamente ninguna actividad invisible, es posible inferir su ocurrencia. Para un mejor entendimiento considérese que una correcta interpretación de la situación de *Actividades invisibles contra lazos* sería el modelo representado en la Figura 1.6 b), y así respectivamente sucede con cada uno de los casos analizados.

Van der Aalst y otros. Los autores fueron los desarrolladores del conocido algoritmo Alpha. El principal aporte fue que garantizaron una clase de modelo que permitía de manera efectiva el trabajo en el área. Esa clase de modelo se formalizó como SWF-nets (siglas de Structured Workflow Nets). El algoritmo Alpha se basa en cuatro tipos de relaciones binarias: *secuencial*, *causal*, *paralelismo* y *sin relación*. La relación de secuencia es la básica a partir de la cual se infieren las demás. Existe una secuencia cuando al menos en una instancia del proceso la actividad A es seguida directamente por B. La relación causal surge cuando A es seguida directamente por B y B no antecede a A directamente. Si A es seguida por B y B es seguida por A entonces se establece una relación de paralelismo. Si A y B no están envueltos en una relación de secuencia entonces la relación es: sin relación.

El algoritmo Alpha no maneja tareas duplicadas. Extensiones realizadas a este algoritmo permitieron manejar correctamente lazos cortos, considerar tareas no atómicas, y de manera general mejoraron las relaciones binarias y las nociones de completitud de las trazas. Hay que resaltar que el algoritmo mina las trazas a partir de la concepción de que las trazas son completas y libres de ruido [2, 18, 26, 93, 94].

Van der Aalst ha trabajado también en otras investigaciones referenciadas en este trabajo. Entre sus últimas investigaciones se debe resaltar el trabajo [17]. En este trabajo

se propone un algoritmo para el descubrimiento de un modelo de proceso que presente un balance adecuado entre generalización y especificación. Es posible cubrir algunos aspectos relacionados con la completitud de los casos pero no es posible resolver todos los problemas asociados con la ausencia de información. Se manejan las actividades invisibles en la *Situación de salto* y la *Situación de división/unión*.

Weijters y otros. La investigación de estos autores puede verse como una extensión del algoritmo Alpha. Para establecer las relaciones de secuencia se emplea la frecuencia de aparición de dichas relaciones en las trazas analizadas. Así se favorecen las relaciones entre las actividades que se manifiestan con mayor frecuencia y se desechan las menos frecuentes. Para identificar los patrones de control de flujo XOR y AND se realiza una reproducción de las trazas sobre el modelo descubierto. El modelo es representado mediante la notación Redes Causales (en ingles C-Net). La implementación de dicho algoritmo se realizó en la herramienta Little Thumb y también en ProM como el plugin Heuristics miner. El algoritmo es robusto ante el ruido [92, 121, 122].

Dongen y otros. Los autores introducen como resultado de la minería de las trazas un modelo en EPCs. El algoritmo desarrollado se estructura en dos pasos y en el primero de estos se obtiene un modelo de proceso por cada una de las trazas analizadas. En este paso se asegura que cada actividad aparezca solo una vez en la traza, a cada instancia de una actividad en la traza se le asigna un identificador único. Ya en el segundo paso se mezclan los modelos obtenidos en el paso anterior y se definen relaciones de división y unión en sus diferentes variantes: AND, OR y XOR [95, 123, 124]. El algoritmo no es robusto ante el ruido.

Wen y otros. El trabajo de estos autores estuvo dirigido a la extensión del algoritmo Alpha. Las primeras mejoras se realizaron en el algoritmo Beta, el cual fue

implementado como el plugin Tsinghuaalpha en la herramienta ProM [79]. Lo distinguía el tratamiento de las actividades no atómicas y el uso del tiempo de ejecución de las actividades para determinar las relaciones de paralelismo y los lazos cortos. Una posterior mejora se implementó como el algoritmo Alpha++. Este incluía otras mejoras como el tratamiento del constructor de no-libre-selección. El algoritmo no es robusto ante el ruido [38, 109].

Medeiros y otros. El principal aporte de estos autores estuvo en la aplicación de algoritmos genéticos en el área de la minería de proceso. El resultado de la aplicación de este tipo de técnica garantizó el manejo de todos los constructores estructurales en el caso del algoritmo GA, exceptuando las tareas duplicadas. El algoritmo DGA es una extensión de GA que permite manejar las tareas duplicadas. El principal inconveniente de estos algoritmos es el tiempo de ejecución. Los algoritmos son robustos ante el ruido [23].

Las pruebas efectuadas utilizando la implementación realizada sobre ProM [79, 107] arrojaron en la mayoría de las situaciones antes expuestas que el modelo obtenido puede interpretarse satisfactoriamente. Se puede inferir que existe ausencia de información, aunque esto no se hace en ningún caso de manera explícita aun cuando existe evidencia que la denota.

Bergenthum y otros. Los autores emplearon métodos de síntesis basados en región para construir Redes de Petri. Para ello se consideran las trazas como un lenguaje finito y el modelo obtenido es una red minimal que recoge el comportamiento del lenguaje dado. Los métodos existentes para la síntesis basada en región se adaptaron para satisfacer las necesidades del área de la minería de proceso.

El algoritmo trata de evitar el comportamiento adicional, es decir, trata de reflejar de la manera más precisa el comportamiento reflejado en las trazas. En consecuencia el algoritmo no es robusto ante el ruido [39].

Rubin y otros. Estos autores aplican técnicas de minería de proceso en el área de desarrollo de software. La fuente de información en este caso no fueron las trazas registradas por los sistemas, sino los documentos e información de manera general almacenados durante el proceso de desarrollo de software en los repositorios de información. Los autores introducen como resultado de la minería de las trazas un modelo en Transition System (TS).

El algoritmo desarrollado se estructura en dos pasos. En el primer paso se genera el modelo en TS. Para obtener el modelo se transita por una fase de pre-procesamiento en la que se estructuran las trazas necesarias para el proceso de minería. Posteriormente se definen los posibles estados y transiciones derivados de los eventos almacenados en las trazas y se construye el modelo. A dicho modelo se le aplican un grupo de estrategias de modificación que permiten obtener un modelo que cubre el comportamiento expresado en las trazas y elimina los lazos. Un segundo paso permite generar una red de Petri a partir del modelo en TS ya descubierto, este paso se basa en la teoría de la región. El algoritmo es capaz de manejar diferentes niveles de abstracción lo cual facilita la adaptabilidad. El algoritmo no es robusto ante el ruido [96].

Günther y otros. Estos autores dirigieron su atención al minado de trazas que reflejan el comportamiento de procesos poco estructurados. En consideración refieren que se deben resaltar en estos casos los aspectos importantes del comportamiento analizado y ocultar lo que puede no ser significativo. Para ello se desarrollaron dos métricas que guían el proceso de descubrimiento del modelo de proceso. La primera de las métricas

es *significación*, la cual permite medir la importancia relativa que tiene un evento y las relaciones de precedencia binarias que existen con respecto a él. Esta especifica el nivel de interés que se puede tener sobre un evento o la ocurrencia de este después de otro. Un aspecto a considerar puede ser la frecuencia de aparición de un evento, mientras mayor es la frecuencia, mayor es la significación del evento. La otra métrica es la *correlación*, determinada por la medida en la que una relación de precedencia entre los eventos puede ser relevante. Permite tener una medida de cuán estrechamente relacionados están dos eventos. Basado en estas dos métricas el algoritmo estructura el modelo de proceso. El algoritmo es robusto ante el ruido [31].

El algoritmo fue implementado como el plugin Fuzzy Miner e incorporado a la herramienta ProM [79].

Farkhady y Aali. Estos autores dirigieron su atención al minado de trazas que están afectadas por el ruido y la completitud. Para ello desarrollaron métricas que permiten definir el tipo de relación que se establece entre las actividades. Con respecto a la completitud se dirige la atención a situaciones en las que se manifiestan actividades en paralelo o de manera concurrente. Es posible cubrir algunos aspectos relacionados con la completitud de los casos pero no es posible resolver todos los problemas asociados con la ausencia de información. Se manejan las actividades invisibles en la *Situación de salto* y la *Situación de división/unión* [33].

Werf y otros. Estos autores proponen un método de descubrimiento basado en Programación Lineal Entera (ILP por sus siglas en inglés). La idea principal es adicionar lugares en la red de Petri resultante, para así restringir las posibles secuencias de disparo. De ahí que se busque cuantos lugares sea posible, de manera que la red de Petri resultante sea consistente con el registro de evento. La conocida teoría de las

regiones resuelve un problema similar, pero para los lenguajes finitos. Este trabajo se apoya en investigaciones como [39, 125]. Primero se define un criterio de optimalidad que permite transformar un sistema de inecuaciones en una ILP. Al resolver el sistema de inecuaciones se obtienen los lugares que conforman la red de Petri. El criterio de optimalidad permite potenciar los lugares en el modelo descubierto que tienen pocas entradas y muchas salidas, lo cual da mayor expresividad al modelo. Por otra parte, se utiliza la relación de causalidad enunciada en el algoritmo Alpha [38, 109], estas relaciones permiten restringir la búsqueda de los lugares necesarios. Además, el tamaño de la red de Petri construida es independiente del número de eventos en el registro de evento, lo que hace a este enfoque más aplicable en situaciones prácticas [40, 126]. El algoritmo desarrollado se implementó como parte de la herramienta ProM [79, 107].

La Tabla 1.2 muestra los resultados de evaluar algunos algoritmos de descubrimiento implementados en la herramienta ProM. El asterisco indica las situaciones que los algoritmos tratan correctamente.

El resumen presentado en la tabla muestra claramente la importancia de resolver la ausencia de información en la etapa de pre-procesamiento y así lograr una mejora en los modelos obtenidos por los diferentes algoritmos de descubrimiento.

1.4 La ausencia de información y el pre-procesamiento de las trazas

Se han desarrollado un grupo de técnicas que durante la etapa de pre-procesamiento del registro de evento permiten detectar las afectaciones asociadas al ruido. En este sentido se puede resaltar algunas de las técnicas de visualización del registro de evento desarrolladas. Estas tienen como objetivo realizar un diagnóstico del proceso para poder identificar los aspectos generales del proceso, posibles anomalías, desviaciones y patrones interesantes.

Tabla 1.2. Algoritmos de descubrimiento de procesos y manejo de la ausencia de información.

Algoritmo	Situación de salto	Situación de división/unión	Actividades invisibles contra Actividades duplicadas	Actividades invisibles contra lazos	Actividades invisibles contra sincronización	Lazos contra actividades invisibles junto a actividades duplicadas
Heuristics miner (Weijters, 2003)	*		*			
Alpha (Van der Aalst, 2003)				*		
Multi-phase (Dongen, 2004)	*	*	*	*		
DWS mining (Grecco, 2005)	*		*	*		
Alpha ++ (Wen, 2006)			*	*	*	
Genetic algorithm (Medeiros, 2006)	*	*	*	*	*	*
Transition System (Rubin, 2007)	*	*			*	
Región Miner (Bergenthum, 2007)	*	*				
Fuzzy Miner (Günther, 2007)	*	*	*	*		
ILP Miner (Werf, 2010)			*	*	*	*

Ejemplos de este tipo de técnica lo constituyen Dotted chart analysis [127] y Trace alignment [84]. Aun cuando sea posible, preferentemente utilizando Trace alignment, identificar *manualmente* algunas de las situaciones más sencillas en las que se manifiesta la ausencia de información, esto solo sería posible en registros de eventos pequeños o medianos (registro de evento que contiene hasta cientos de casos).

Recientemente, se desarrolló un trabajo que realiza una transformación y limpieza semántica de los datos que forman el registro de evento [32]. Este trabajo está dirigido a tratar con las afectaciones asociadas al ruido. Para la limpieza se definen un grupo de restricciones en correspondencia con un dominio en específico y se identifican las situaciones poco frecuentes. Las violaciones detectadas posibilitan que los casos afectados se eliminen del registro de evento. Haciendo un análisis de esta técnica se determina que, aun cuando pudiesen definirse restricciones asociadas a las situaciones en la que se manifiesta la ausencia de información, especialmente las más sencillas de reconocer como, *Situación de salto, Situación de división/unión, Actividades invisibles contra Actividades duplicadas, Actividades invisibles contra lazos*; esto tendría que hacerse para cada proceso analizado y considerando los diferentes niveles de abstracción en los que se pueden manifestar la ausencia de información. Además, la investigación no posibilita estimar las actividades invisibles en correspondencia con las situaciones posiblemente identificadas.

LTL Checker es una técnica desarrollada por Van der Aalst (2005) que presenta las mismas limitantes que la técnica antes mencionada [19]. Con respecto a las restricciones que son posibles definir utilizando linear temporal logic, en el caso de LTL Checker, es posible hacerlo exclusivamente a nivel de actividades, lo que limita aun más su aplicabilidad en las situaciones relacionadas con la ausencia de información.

Ninguna de las técnicas analizadas tiene como propósito la detección de las situaciones en las que se manifiesta la ausencia de información abordada en este trabajo y tampoco realizan estimaciones de la información ausente. Pero es significativo que estas técnicas proponen realizar un análisis en la etapa de pre-procesamiento de los registros de

eventos, lo cual puede ser útil para realizar la estimación de la información ausente. De esta forma se facilita la posterior aplicación de técnicas de descubrimiento.

Para esta investigación la estimación de información ausente se define como:

Definición 1.3 (Estimación de información ausente): La estimación de información ausente es el proceso mediante el cual se transforma el conjunto de trazas $\mathbb{T} = \{T_1, T_2, ..., T_n\}$ en otro conjunto de trazas $\check{\mathbf{T}} = \{\check{T}_1, \check{T}_2, ..., \check{T}_n\}$ donde, $\check{T}_i \in (\Sigma \cup \Lambda)^+$ para $1 \leq i \leq n$, Σ es el conjunto de todas las actividades y $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_w\}$ es el conjunto de actividades invisibles estimadas. $| T_i | \leq | \check{T}_i |$ ■

Esta definición está relacionada con la etapa de pre-procesamiento dado que se considera que debido a la estimación se obtiene un nuevo registro de evento con la información estimada y no un modelo descubierto con dicha información.

1.5 Métricas para evaluar las afectaciones en la estructura y compresión

La ausencia de información en el registro de evento provoca que en el modelo descubierto se establezcan incorrectas relaciones entre las actividades incorporadas a este. Por lo cual es necesario medir el grado en el que el modelo descubierto representa el comportamiento observado en el registro de evento. Este tipo de verificaciones se enmarcan en el área de Chequeo de conformidad [13, 20, 21, 36], para la cual han sido desarrolladas un conjunto de métricas. Las métricas se han agrupado considerando varias dimensiones, por ejemplo el Fitness, la Precisión, la Generalidad y la Estructura [128-130]. La mayoría de las métricas desarrolladas consideran que el modelo de proceso está representado por una red de Petri, aun cuando la métrica puede generalizarse para notaciones con semejante expresividad.

Es necesario conocer hasta qué punto la ausencia de información afecta la estructura y comprensión del modelo descubierto. Para ello se han analizado recientes estudios realizados [128, 131] que fundamentan la selección de determinadas métricas que se ajustan a las características de esta investigación. Cuando se hace referencia a las afectaciones estructurales en el modelo se entiende como las afectaciones ocasionadas por las incorrectas relaciones que se establecen en el modelo a causa de la ausencia de información. Estas afectaciones no se relacionan con la dimensión Estructura antes mencionada.

Para medir las afectaciones que provoca la ausencia de información sobre la estructura del modelo descubierto se seleccionan dos métricas asociadas al Fitness, específicamente Fitness Unsatisfied y Fitness Unhandled [128]. Ambas métricas permiten medir la afectación que provocan las incorrectas relaciones que se establecen en el modelo descubierto a partir del registro de evento con ausencia de información. Es necesario señalar que en la mayoría de los trabajos relacionados con el chequeo de conformidad se hace referencia a la métrica Fitness que incorpora las dos antes mencionadas. Sin embargo, en este trabajo se propone su uso de manera fragmentada para ganar en precisión durante la medición.

Fitness Unsatisfied: Indica cuán bien un modelo se corresponde con un registro de evento, penalizándose el comportamiento que no se refleja en el modelo y que aparece en el registro de evento.

Fitness Unhandled: Se penaliza en correspondencia con la cantidad de tokens (considerando que el modelo está representado por una red de Petri) que quedan a la izquierda del modelo al completar la reproducción de una traza. Indicando esto el comportamiento que no se pudo manejar en el modelo.

Para medir las afectaciones que provoca la ausencia de información sobre la comprensión del modelo descubierto se seleccionan las métricas asociadas al Fitness y además se propone aplicar dos métricas asociadas a la medición de la Precisión, específicamente Precision y Non Fit Traces. Ambas métricas están contenidas en la métrica ETCPrecision [131]. La propuesta de medir la precisión del modelo se realiza considerando que un modelo con un alto grado de precisión facilita su comprensión.

Precision: permite evaluar la precisión de un modelo en correspondencia con el comportamiento observado en un registro de evento. Penaliza el comportamiento extra reflejado en el modelo. En el presente trabajo le llamaremos a esta métrica ETCPrecision.

Non Fit Traces: Cuantifica las trazas que no pudieron ser representadas correctamente por el modelo.

Para favorecer la comprensión del modelo descubierto no deben existir en éste problemas asociados a la dimensión de Estructura. En este sentido se puede evaluar el modelo mediante la métrica Improved Structural Appropriateness [132]. Esta métrica penaliza las actividades invisibles redundantes y las actividades duplicadas alternativamente.

1.6 Detección de patrones recurrentes en las trazas

Para poder estimar la información ausente se deben identificar las situaciones en las que esta se manifiesta en las trazas. Para ello es necesario un mecanismo que identifique la manifestación de los patrones representativos de dichas situaciones en las trazas.

A partir de la identificación de las situaciones de ausencia de información se puede hacer la estimación de información correspondiente. Es necesario que este proceso tenga lugar antes de la etapa de descubrimiento, específicamente durante la etapa de

pre-procesamiento de las trazas. La estimación de las actividades invisibles da como resultado un nuevo registro de evento que puede ser utilizado por cualquiera de las técnicas de descubrimiento del modelo de proceso existentes.

La detección de patrones interesantes es parte del diagnóstico del proceso [84]. Varias han sido las técnicas que posibilitan la detección de patrones interesantes en trazas [43, 133-135], sin embargo estas técnicas presentan problemas como:

- Se detectan una gran cantidad de patrones que pueden ser insignificantes. Es necesario puntualizar que existen métricas para filtrar los patrones menos significativos [136].
- Se identifican patrones recurrentes pero no es posible correlacionar los patrones detectados. Esto representa un problema debido a que los patrones detectados constituyen secuencias de actividades contenidas en una o varias trazas y en la mayoría de las situaciones expuestas la ausencia de información se manifestaba en varias trazas.
- No es posible tener una vista general del proceso para enmarcar claramente los patrones detectados en un contexto.

Este último problema puede ser sumamente importante debido a que es necesario saber en qué contexto se estima una actividad invisible para poder posteriormente reflejarlo de manera coherente en el nuevo registro de evento. En consecuencia se plantea que es útil identificar los subprocesos en los que se descompone el proceso analizado y la relación existente entre dichos subprocesos.

Definición 1.4 (Subproceso): Un subproceso es una encapsulación de actividades del negocio que representan una compleja y lógica unidad de trabajo. Los subprocesos tienen sus propios atributos y metas, pero contribuyen a la meta del proceso que los

contiene. Un subproceso es también un proceso y su mínima expresión es una actividad.■

Entre las técnicas analizadas se encuentra Trace alignment, la cual facilita la detección de los patrones recurrentes [41, 84]. También permite tener una visión holística de las trazas lo que favorece la detección de relaciones entre los patrones y su detección asociados a un contexto. Considerando estos aspectos se puede plantear que esta técnica representa un punto de partida sobre el cual apoyarse para la detección de información ausente.

1.7 Alineación de trazas

La alineación de las trazas es una técnica que utiliza la programación dinámica para tabular las trazas de forma tal que se simplifiquen los problemas asociados al entendimiento de las relaciones entre las actividades. La técnica está inspirada en la Alineación de Múltiples Secuencias (por sus siglas en inglés MSA) [137-140]. La alineación de múltiples secuencias es una herramienta fundamental en el área de la bioinformática.

Todavía existen problemas por resolver en la alineación de secuencias tales como: tratamiento de conjuntos de datos cada vez más grandes y complicados, incorrectas alineaciones, obtención de alineaciones precisas de secuencias no codificadas o no transcritas, entre muchos otros.

En recientes trabajos se logró adaptar esta última técnica a la alineación de trazas [41, 84]. La alineación de secuencias biológicas comúnmente ocurre sobre secuencias homogéneas y con poca variación en el tamaño de las secuencias [141]. En la minería de proceso se manejan secuencias de actividades, estas constituyen los casos o trazas que conforman un registro de evento. Los casos no necesariamente se manifiestan de

manera homogénea y pueden presentar variaciones en el tamaño. Las matrices de puntuación para la alineación de trazas se deben definir automáticamente a partir del registro de evento o son proporcionadas por los expertos del negocio.

Por otra parte, las secuencias biológicas tienen un alfabeto de longitud cuatro (para cuatro ácidos nucleicos) o 20 (para los aminoácidos), mientras que el número de actividades distintas (clases de eventos) en un registro de evento en la minería de proceso puede ser del orden de unos pocos cientos. A esto se le suma la complejidad de derivar una correcta matriz de puntuación y la alineación de las trazas.

La calidad de la alineación final sobre un conjunto de trazas depende en gran medida del orden en que se realiza la alineación progresiva. En el trabajo desarrollado por Bose y Van der Aalst se utilizó el algoritmo Agglomerative Hierarchical Clustering para agrupar las trazas y así garantizar una mayor homogeneidad en el grupo de trazas a procesar en las primeras iteraciones de la alineación progresiva. Existen diferentes enfoques de agrupamiento sensibles al contexto que permiten agrupar las trazas de forma coherente [29, 133, 142].

Para una mayor comprensión de la alineación de trazas se expone su formalización (tomado de [41]).

Definición 1.5 (Alineación de las trazas): La alineación de las trazas sobre un conjunto de trazas $T = \{T_1, T_2, \ldots, T_n\}$ se define como el mapeo del conjunto de trazas de T sobre otro conjunto de trazas $\overline{T} = \{\overline{T}_1, \overline{T}_2, \ldots, \overline{T}_n\}$ donde cada $\overline{T}_i \in (\Sigma \cup \{-\})^+$ para $1 \leq i \leq n$

- Existe un $m \in N$ tal que $|\overline{T}_1| = |\overline{T}_2| = \ldots = |\overline{T}_n| = m$
- $\overline{T}_i$ es igual a Ti después de eliminar todos los símbolos "–"
- No existe una $k \in \{1, \ldots, m\}$ tal que $\forall\ 1 \leq i \leq n$, $\overline{T}_i(k) = -$. ■

En la definición dada *m* representa la longitud de la alineación. Una alineación sobre un conjunto de trazas puede ser representada por una matriz rectangular $\mathcal{A} = \{a_{ij}\}$ ($1 \leq i \leq n$, $1 \leq j \leq m$) sobre $\Sigma' = \Sigma \cup \{-\}$ donde "–" denota un vacío. La tercera condición de la definición implica que no existe una columna de $\mathcal{A}$ que contiene solo símbolos "–'. Es necesario resaltar que existen varias posibles alineaciones sobre un conjunto de trazas y que la longitud de la alineación, *m*, satisface la relación $l_{max} \leq m \leq l_{sum}$ donde l_{max} es la máxima longitud de una traza contenida en T y l_{sum} es la suma de las longitudes de todas las trazas contenidas en T.

Al obtener una alineación de las trazas se pueden determinar patrones interesantes como secuencias entre bloques de eventos, lo cual resulta útil en la descomposición del proceso analizado. También es posible detectar actividades que se realizan sincrónicamente.

1.8 Conclusiones del capítulo

En el tratamiento de la ausencia de información se han identificado dos aristas, una se desarrolla en la etapa de pre-procesamiento del registro de evento y la otra durante el descubrimiento del modelo de proceso.

Los algoritmos de descubrimiento obtienen modelos diferentes ante situaciones similares de ausencia de información. Esta ambigüedad en la interpretación de las trazas está determinada por los constructores de control de flujo que puede manejar cada algoritmo y la forma en que lo hace.

Los trabajos analizados y que se enfocan en el pre-procesamiento de las trazas tienen como limitante que, aun cuando permiten la detección de las manifestaciones del ruido, no están dirigidos a identificar las situaciones de ausencia de información descritas, en consecuencia no permiten la estimación de las actividades invisibles.

Se hace necesario que como parte de la etapa de pre-procesamiento de las trazas se considere la estimación de información ausente en las trazas para así garantizar la obtención de modelos mucho más ajustados al comportamiento real del proceso de la empresa analizada.

CAPÍTULO 2. MODELO PARA LA ESTIMACIÓN DE INFORMACIÓN AUSENTE

En este capítulo se abordan dos manifestaciones de ausencia de información que no habían sido reportadas en la literatura y se expone un modelo para la estimación de información ausente en las trazas usadas en la minería de proceso. Se describe la fundamentación del modelo propuesto, los componentes que lo conforman, así como la relación que se establece entre estos componentes.

2.1 Manifestaciones de ausencia de información detectadas

Además de las situaciones expuestas en el anterior capítulo (sección 1.2) hay otras circunstancias en las que pudiese existir ausencia de información, sin embargo, no son "significativas" al no afectarse la estructura del modelo descubierto [36]. Por ejemplo, cuando en un proceso existe una secuencia de actividades y alguna de ellas no queda reflejada en las trazas.

Existen otras circunstancias en las que se evidencia ausencia de información y la afectación en la estructura del modelo descubierto es "significativa", a pesar de ello, estas manifestaciones no han sido tomadas en consideración por los algoritmos de descubrimiento consultados.

El autor del presente trabajo durante su investigación detectó dos manifestaciones de ausencia de información que no habían sido reportadas en la literatura consultada. La descripción de estas y un ejemplo de la afectación que provoca en los modelos descubiertos se exponen a continuación.

Considérese un proceso conformado por dos subprocesos ordenados secuencialmente, en el que las actividades de fin del primer subproceso e inicio del segundo no se reflejan en las trazas. En este caso al no quedar claramente delimitados los subprocesos se establece una o

varias relaciones de causalidad entre las actividades que conforman ambos subprocesos cuando en la realidad esto no sucede. A esta nueva situación identificada por el autor del presente trabajo se le ha denominado *Secuencia oculta de subprocesos*, lo que ayuda a la comprensión de posteriores valoraciones.

Un ejemplo de la situación *Secuencia oculta de subprocesos* se muestra a continuación.

Las secuencias de actividades BCXYDLI, BOXYLDI, BWXYDLI se corresponden de manera correcta con el modelo que se muestra en la Figura 2.1.

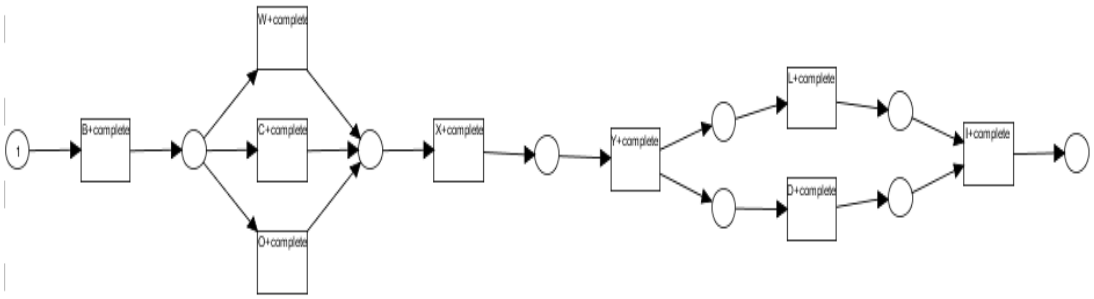

Figura 2.1. Red de Workflow, modelo descubierto usando el algoritmo Alpha.

En el modelo se pueden identificar dos subprocesos ordenados secuencialmente. El primer subproceso se inicia en la actividad B y termina en la actividad X. El segundo subproceso se inicia en Y y termina en la actividad I. Si las actividades X y Y son actividades invisibles el modelo descubierto utilizando el algoritmo Alpha [18] quedaría como se muestra en la Figura 2.2. El modelo refleja incorrectas relaciones de causalidad entre las actividades. El modelo que se muestra en la Figura 2.2 presenta los siguientes problemas:

- La actividad I no se puede habilitar debido a que requiere dos tokens de entrada y estos no es posible producirlos simultáneamente.
- Se establecen relaciones de causalidad incorrectas entre las actividades W y D, C y D, O y D.

Estos problemas se manifestaron como consecuencia de la ausencia de información. Específicamente, las causas están en la ausencia de actividades que permitan definir el fin del subproceso (en este caso X) y el inicio del que le sigue secuencialmente (en este caso Y).

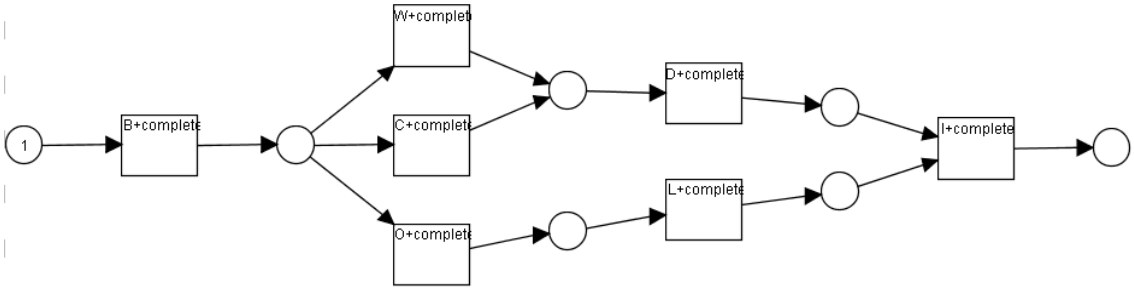

Figura 2.2. Red de Workflow, modelo descubierto usando el algoritmo Alpha.

Otra manifestación de ausencia de información no reportada en la literatura puede detectarse si se tienen en cuenta las frecuencias relativas de ocurrencia de las opciones a seleccionar. Un ejemplo de esta situación es el proceso (P_0) representado en la Figura 2.3. Las actividades B y F no se encuentran informatizadas, y por tanto, no parecen en las trazas almacenadas.

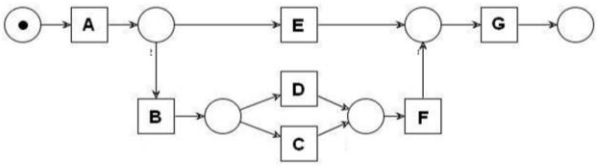

Figura 2.3. Red de Workflow, representa el proceso P_0.

Una muestra representativa de las trazas almacenadas se representa en la Tabla 2.1. Cada clase, C_1, C_2 o C_3, se corresponde con una secuencia única de actividades. A iguales secuencias de actividades les corresponde una única clase.

Tabla 2.1. Representación de los casos.

Caso	Secuencia de tareas	Clase
1	AEG	C_1
2	AEG	C_1
3	ADG	C_2
4	ACG	C_3

A partir de las trazas reflejadas en la Tabla 2.1 los algoritmos de descubrimiento de proceso consultados descubrirán un modelo de proceso (P_1) equivalente al reflejado en la Figura 2.4. P_1 difiere significativamente de P_0, afectándose por la ausencia de información la estructura del modelo de proceso descubierto.

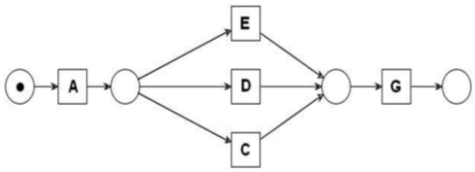

Figura 2.4. Red de Workflow, representa el proceso P_1.

Si se analizan las clases se puede percibir que $|C_1|$ representa el 50 % del total de casos, mientras que $|C_2|$ representa el 25 % del total de casos al igual que $|C_3|$. Esta distribución indica la ausencia de las actividades B y F, debido a que, si E, D y C son opciones equiprobables se debía cumplir que $|C_1| \approx |C_2| \approx |C_3|$. De esta misma forma pudiese ilustrarse otro grupo de ejemplos en los que bajo las mismas condiciones puede encontrarse evidencia de la ausencia de información. A este tipo de situaciones se denomina por el autor del presente trabajo: *Opciones equiprobables*.

El análisis desde esta arista es desechado habitualmente por las técnicas de descubrimiento analizadas, debido a que no se cumplen las condiciones antes expresadas por la propia naturaleza de las instancias del proceso. Sin embargo, para determinados procesos ante una situación de selección es conocida la frecuencia relativa de ocurrencia de cada una de las opciones y se puede partir del supuesto teórico expresado anteriormente, lo cual puede ser empleado también para detectar y estimar posible información ausente. El objetivo de este análisis es ilustrar cómo, de suceder una situación como esta, no se toma en cuenta, durante el descubrimiento del modelo de proceso, como evidencia de la ausencia de información.

2.2 Fundamentos de un modelo para la estimación de información ausente en las trazas

Como se había reflejado en el primer capítulo (secciones 1.3 y 1.4) existen dos posibles enfoques en el tratamiento de la ausencia de información. Resolver este tipo de problema desde el enfoque del descubrimiento de un modelo de proceso requiere desarrollar una nueva técnica que trate todas las manifestaciones de ausencia de información enunciadas y además sea capaz de satisfacer requerimientos asociados al descubrimiento, tales como, la identificación de los patrones de flujo de trabajo y el tratamiento del ruido y la desestructuración. Es por ello que, resulta conveniente dirigir la atención a la estimación de información ausente durante la etapa de pre-procesamiento de las trazas, de forma que las técnicas de descubrimiento desarrolladas utilicen como punto de partida un registro de evento con la información estimada. De esta forma se evita la ambigüedad en la interpretación de las trazas, se favorece la obtención de un modelo en el cual no se establecen incorrectas relaciones entre las actividades y se facilita su comprensión en situaciones en las que existe ausencia de información.

La estimación de información ausente es un fenómeno complejo debido a que es necesario que el analista tenga en cuenta las características del proceso para determinar qué situaciones pueden interpretarse como ausencia de información. Las actividades invisibles pueden manifestarse considerando las diferentes situaciones enunciadas en este trabajo, además hay que considerar que estas manifestaciones pueden combinarse y reflejarse a diferentes niveles de abstracción lo cual tributa a la complejidad del fenómeno. En correspondencia con lo planteado es necesario que el analista del proceso cuente con un modelo para la estimación de la información ausente en las trazas usadas en la minería de proceso.

En esta investigación se propone un modelo que constituye una representación teórica del fenómeno de estimación en el contexto enunciado y que describe los elementos

fundamentales que caracterizan el fenómeno, así como, las relaciones entre estos. El modelo propuesto debe ser aplicado en la etapa de pre-procesamiento de las trazas y permitir identificar las manifestaciones de ausencia de información descritas en el capítulo uno (sección 1.2) y en el capítulo dos (sección 2.1) y estimar las actividades invisibles correspondientes, sin importar la combinación de estas manifestaciones y el nivel de abstracción en el que ocurran.

2.3 Aspectos generales del modelo propuesto

El modelo propuesto se le denominó EIAT (Estimación de Información Ausente en las Trazas). Este se aplica en la etapa de pre-procesamiento de las trazas, por lo que, tiene como entrada un registro de evento y tiene como salida un registro de evento que incorpora la información estimada, ambos registros de evento con un formato XES o MXML.

El modelo está compuesto por cinco componentes:

1. Componente para la alineación de las trazas
2. Componente para pre-procesar la alineación
3. Componente para determinar el árbol de bloques de construcción
4. Componente para la aplicación de los operadores de ausencia de información
5. Componente para construir el registro de evento con información estimada

El esquema de la Figura 2.5 permite tener una visión general del modelo EIAT. En el esquema se utiliza AI como abreviatura de Ausencia de Información e IE como Información Estimada.

La estimación de la información ausente en un registro de evento requiere la realización de tres acciones básicas, la primera es descomponer en subprocesos el proceso reflejado en las trazas, de forma tal que, en un segundo paso se puedan identificar las manifestaciones de

ausencia de información enunciadas sin importar el grado de abstracción en el que se produzcan.

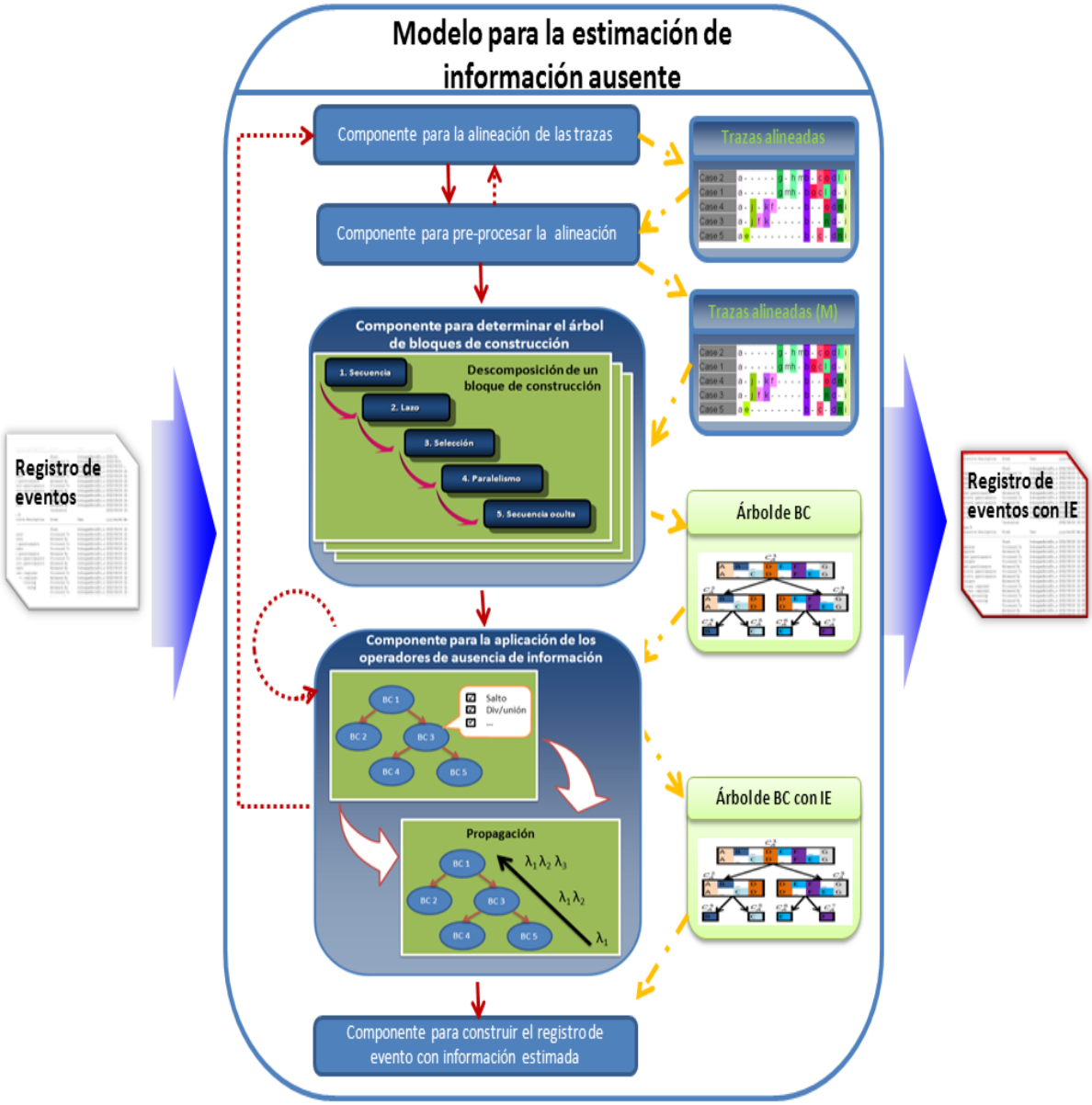

Figura 2.5. Esquema representativo del modelo EIAT.

En correspondencia los tres primeros componentes se encargan de la descomposición del proceso en subprocesos. Para ello inicialmente se emplea la alineación de las trazas, que como se había descrito en el capítulo uno (sección 1.6), facilita la detección de los patrones de flujo de trabajo. De la descomposición del proceso se encargan los componentes dos y tres, el dos propicia la corrección de la alineación mientras que el tres se encarga de la

descomposición. La tercera acción está dirigida a estimar la información ausente en correspondencia con las manifestaciones identificadas y las características del subproceso analizado. El cuarto componente se encarga de la segunda y la tercera acción. Finalmente considerando la información estimada se genera mediante el quinto componente un nuevo registro de evento.

El orden en el que se van empleando los componentes se refleja en el esquema de la Figura 2.5 mediante las flechas que los unen. Las flechas de puntos indican que se puede volver a aplicar componentes anteriormente utilizados en dependencia del resultado obtenido. Las entradas y salidas de cada componente se indican con flechas discontinuas.

Como parte de la descripción de cada componente se proponen un grupo de algoritmos desarrollados por el autor del presente trabajo que facilitan la aplicación del modelo EIAT. Para el cálculo de la complejidad temporal se considera en cada algoritmo a *n* y *m* como la cantidad de filas y columnas respectivamente de la matriz que representa el bloque de construcción analizado.

2.4 Principios del modelo EIAT

El modelo EIAT cumple con un conjunto de principios que han sido definidos en el Manifiesto sobre minería de proceso [14], los cuales tienen como propósito guiar el trabajo en esta área del conocimiento.

Principio 1: Los datos de los eventos deberían ser tratados como ciudadanos de primera clase.

El modelo propuesto permite el tratamiento de la ausencia de información en el registro de evento lo cual propicia un mejor aprovechamiento de dicho modelo.

Principio 2: La extracción de los registros de evento debería ser impulsada por preguntas.

La solución propuesta en esta investigación se aplica en la etapa de pre-procesamiento por lo cual durante la extracción del registro de eventos permite responder a las siguientes interrogantes: ¿Existe ausencia de información en el registro de evento?, ¿En qué contexto se manifiesta la ausencia de información?, ¿Qué manifestaciones de ausencia de información están presentes en el registro de evento?

Al responder estas interrogantes se puede iterativamente ir ajustando la información necesaria para la conformación del registro de evento definitivo.

Principio 3: Se debe dar soporte a la concurrencia, elección y otros conceptos básicos de control de flujo.

El modelo propuesto durante la descomposición del proceso permite obtener una representación que da soporte a la concurrencia, elección y otros conceptos básicos de control de flujo.

Principio 4: Los eventos deben estar relacionados a elementos del modelo.

Este principio busca una mayor correspondencia entre lo que se refleja en el registro de evento y el modelo de proceso que lo representa. El modelo propuesto permite obtener un registro de evento con la información estimada lo que disminuye las afectaciones en la estructura y comprensión del modelo de proceso descubierto. Esto propicia que exista una mayor correspondencia entre los eventos que ocurren durante la ejecución del proceso de negocio y el modelo que representa dicho proceso.

Principio 5: Se debe tratar a los modelos como abstracciones útiles de la realidad.

El modelo propuesto durante la descomposición del proceso permite obtener una representación que constituye una abstracción útil de la realidad. Los trabajos [143-145] evidencian la utilidad de la representación obtenida en el diagnóstico del proceso. Además, el modelo de proceso descubierto a partir del registro de evento con la información estimada propicia una abstracción más cercana a la realidad.

Principio 6: La minería de procesos debe ser un proceso continuo.

Este principio se refiere a que la minería de proceso debería aplicarse durante la ejecución del proceso de negocio de forma tal que pueda irse analizando constantemente su evolución. En este sentido el modelo propuesto permite realizar la estimación de la información ausente considerando el registro de evento que se obtiene en cada momento y en correspondencia se favorece el análisis del procesos de manera continua.

2.5 Componente para la alineación de las trazas

Este componente se encarga del agrupamiento de las trazas y su alineación. La obtención de las trazas alineadas favorece la posterior descomposición del proceso, la detección de las manifestaciones de ausencia de información y la estimación correspondiente.

El agrupamiento de las trazas se puede realizar utilizando alguna de las técnicas desarrolladas en este sentido [29, 30, 133]. La alineación de un grupo debe cumplir con la *Definición 1.5*. Como resultado de la alineación de un grupo de trazas se obtiene una matriz $\mathcal{A}$. Para la aplicación del modelo propuesto en este trabajo se recomienda formar un solo grupo que represente la totalidad del proceso analizado.

Para la aplicación del modelo EIAT se propone emplear la técnica de alineación de trazas desarrollada por Bose y Van der Aalst. Los dos trabajos [41, 84] desarrollados por estos autores constituyen los únicos referentes identificados en la literatura.

Las técnicas para la alineación de las trazas propuestas por Bose y Van der Aalst requieren de la configuración de varios aspectos importantes. Estos aspectos están asociados a la definición de las matrices de puntuación, ya sea la correspondiente a la sustitución o a la de inserción y eliminación. Estas matrices pueden ser elaboradas por un experto del negocio analizado o se pueden derivar de la información contenida en las trazas. Para la aplicación del modelo propuesto se pueden emplear cuales quiera de las variantes expuestas.

2.6 Componente para pre-procesar la alineación

Este componente permite corregir algunos aspectos de la alineación de las trazas que pueden constituir un problema para los componentes que se aplican posteriormente. Como entrada el componente tiene la matriz $\mathcal{A}$.

El primero de los aspectos que se corrige está asociado a que en la alineación se pueden manifestar múltiples actividades ocupando una columna. La corrección se enfoca en separar las actividades en columnas independientes y continuas. El orden de las actividades no es significativo para la estimación. El algoritmo propuesto por el autor, que permite identificar las columnas compartidas por varias actividades y la corrección de esta situación se muestra en el Anexo 1.

El algoritmo propuesto tiene dos partes fundamentales, una dirigida a detectar las columnas compartidas por varias actividades y la segunda asociada a la inserción de nuevas columnas para separar las actividades que comparten una columna previamente identificada. De estas dos operaciones, que se realizan secuencialmente, la que mayor complejidad temporal tiene asociada es la primera, dado que para detectar las columnas compartidas es necesario recorrer toda la matriz alineada. En consecuencia, la complejidad temporal del algoritmo que implementa la corrección descrita es O($m \cdot n$).

Como salida de este componente se tiene la matriz $\mathcal{A}'$ construida a partir de la modificación de la matriz $\mathcal{A}$.

A partir de la alineación también se pueden determinar los casos incompletos, es decir, casos que no terminan con las actividades finales identificadas para el proceso. El resultado de la alineación permite detectar visualmente este tipo de casos, dado que presentan vacíos (símbolo de "–") en las columnas correspondientes a las actividades finales del proceso. Estos casos son eliminados del registro de evento si así lo desea el analista.

En el caso de modificarse el registro de evento debe aplicarse nuevamente el *Componente para la alineación de las trazas*.

2.7 Componente para determinar el árbol de bloques de construcción

Este componente permite obtener un árbol de bloques de construcción que representa la descomposición jerárquica en subprocesos del proceso analizado. La matriz $\mathcal{A}'$ constituye la entrada de este componente.

Varios han sido los trabajos publicados o presentados en eventos por el autor de la presente investigación. Estos trabajos están relacionados con la aplicación de las técnicas de minería de proceso en el diagnóstico del proceso analizado, lo cual tributa a la auditoria empresarial. Estos trabajos abordan de una forma u otra la descomposición de un proceso propuesta en esta sección [143-149]. La totalidad de los algoritmos empleados en la descomposición del proceso y que se describen en esta sección constituyen un aporte de esta investigación.

Antes de llegar a una formalización del concepto de *bloque de construcción* es necesario señalar algunos aspectos relacionados con un proceso de negocio y su descomposición en subprocesos.

Un proceso puede descomponerse en varios subprocesos mediante los patrones de flujo de trabajo siguientes:

- Secuencia: dos subprocesos se encuentran ordenados secuencialmente, si inmediatamente después de que ocurra el primer subproceso, ocurre el segundo.
- Selección (XOR u OR): dos subprocesos se encuentran ordenados como opciones de una selección, si en cada caso o instancia del proceso sólo ocurre uno de ellos (XOR) u ocurren los dos en cualquier orden (OR).
- Paralelismo: dos subprocesos se encuentran ordenados en paralelo si ocurren los dos simultáneamente.

- Lazo: un lazo se manifiesta cuando un subproceso se repite en múltiples ocasiones.

Los subprocesos pueden descomponerse en otros subprocesos hasta el nivel de actividad. Esto permite construir un árbol en el que cada nivel tiene menor grado de abstracción.

Definición 2.1 (Bloque de construcción y descomposición en bloques de construcción): Sea *S* el conjunto de todos los subprocesos que componen a un proceso *P*, $\mathcal{L}$ el registro de evento que representa a las instancias del proceso *P* ejecutadas, $\mathcal{A}$ la matriz obtenida a partir de la alineación de las trazas contenidas en $\mathcal{L}$ y $Q_{\mathcal{A}}$ el conjunto de todas las sub-matrices de $\mathcal{A}$.

El conjunto de sub-matrices que representan a los subprocesos de *S* se denota por $Q'_{\mathcal{A}}$, tal que $Q'_{\mathcal{A}} \subseteq Q_{\mathcal{A}}$. Sean $C^i_{\mathcal{A}}$ y $C^j_{\mathcal{A}}$ sub-matrices de $Q'_{\mathcal{A}}$, la relación de secuencia entre dos subprocesos representados por $C^i_{\mathcal{A}}$ y $C^j_{\mathcal{A}}$ se denota por $C^i_{\mathcal{A}} >'_{\mathcal{L}} C^j_{\mathcal{A}}$. De forma análoga se denota la relación de selección por $C^i_{\mathcal{A}} \#'_{\mathcal{L}} C^j_{\mathcal{A}}$ y la relación de paralelismo por $C^i_{\mathcal{A}} ||'_{\mathcal{L}} C^j_{\mathcal{A}}$.

En el caso de manifestarse un lazo, la descomposición de $C^i_{\mathcal{A}}$ se realiza en un único sub-proceso que se repite múltiples veces y se denota por $(C^j_{\mathcal{A}})^*$.

Sea $s_i \in S$ un subproceso representado por la matriz $C^i_{\mathcal{A}} \in Q'_{\mathcal{A}}$ y que está compuesto por una secuencia de subprocesos representados por $C^j_{\mathcal{A}}$, …, $C^{j+k}_{\mathcal{A}}$, entonces tanto la matriz $C^i_{\mathcal{A}}$ como el conjunto $\{C^j_{\mathcal{A}}, \ldots, C^{j+k}_{\mathcal{A}}\}$ se denominan *bloques de construcción* y los sub-procesos representados por $\{C^j_{\mathcal{A}}, \ldots, C^{j+k}_{\mathcal{A}}\}$ se relacionan de una única forma (secuencia, paralelismo, selección o lazo).■

En el *Algoritmo para determinar el árbol de bloques de construcción*, que constituye un aporte del autor de esta investigación, se describen los pasos para determinar el árbol de bloques de construcción. Se emplea una representación arbórea porque muestra la forma en la que el proceso se descompone en otros subprocesos en cada nivel.

Algoritmo 1 Algoritmo para determinar el árbol de bloques de construcción

Entrada: Matriz $\mathcal{A}'$

Salida: Árbol de bloques de construcción

1: Crear un árbol vacío.

2: Crear un bloque de construcción $C^1_{\mathcal{A}}$ y asociar al nodo raíz del árbol la matriz $\mathcal{A}'$, tal que $C^1_{\mathcal{A}} = \mathcal{A}'$.

3: **Si** $C^i_{\mathcal{A}}$ no es una matriz con una sola fila **Entonces**

4: *LH* = *Buscar secuencia*($C^i_{\mathcal{A}}$). *LH* es la lista que almacena los bloques de construcción obtenidos producto de la descomposición realizada.

5: **Si** |*LH*| = 0 **Entonces**

6: LH = *Buscar lazo*($C^i_{\mathcal{A}}$)

7: **Si** |*LH*| = 0 **Entonces**

8: LH = *Buscar XOR-OR*($C^i_{\mathcal{A}}$)

9: **Si** |*LH*| = 0 **Entonces**

10: LH = *Buscar paralelismo*($C^i_{\mathcal{A}}$)

11: **Si** |*LH*| = 0 **Entonces**

12: LH = *Buscar secuencia oculta*($C^i_{\mathcal{A}}$)

FinSi

FinSi

FinSi

FinSi

13: **Para** cada bloque de construcción *i* contenido en la lista *LH* **Hacer**

14: Modificar *i*. Eliminándose de ser necesario, las filas repetidas y las columnas que solo contienen símbolos vacíos ("–")

15: Adicionar *i* como hijo del nodo que contiene a $C^{i}_{\mathcal{A}}$

16: Aplicar el *Algoritmo para determinación del árbol de bloques de construcción* al bloque de construcción modificado comenzando por el paso 3

17: **Si** el árbol obtenido en el paso anterior ≠ Ø **Entonces**

18: Adicionar los nodos hijos de la raíz del árbol obtenido como hijos del nodo que contiene el bloque de construcción *i*

FinSi

FinPara

Sino

19: Devolver un árbol vacío

FinSi

20: Devolver el árbol de bloques de construcción construido

En el *Algoritmo para determinar el árbol de bloques de construcción* la operación que mayor complejidad temporal tiene es *Buscar secuencia oculta* en un bloque de construcción. Dicha complejidad es O($m^4 \cdot n$) y se explica posteriormente cuando se describe el procedimiento. Si consideramos que la mayor cantidad de niveles del árbol de bloques de construcción que se pueden construir utilizando esta operación es *m*-1 entonces la complejidad temporal de este algoritmo es O($m^5 \cdot n$).

En la Figura 2.6 se muestra un ejemplo que facilita la compresión del árbol de bloques de construcción obtenido al aplicar el *Algoritmo para determinar el árbol de bloques de construcción*.

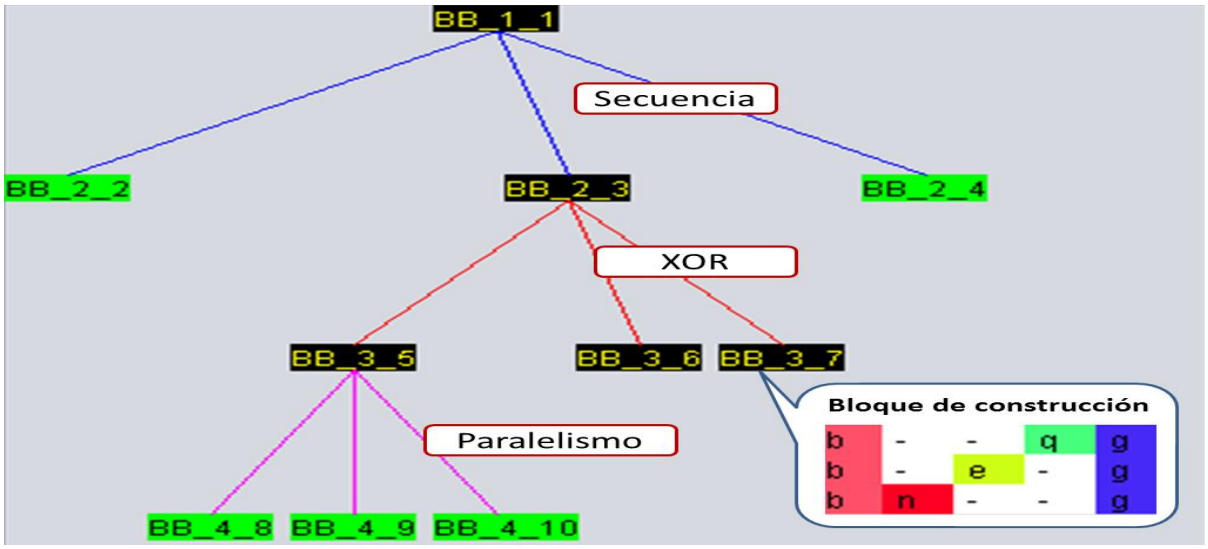

Figura 2.6. Ejemplo de un árbol de bloques de construcción.

A continuación se describen los algoritmos para *Buscar secuencia*, Buscar lazo, *Buscar XOR-OR*, *Buscar paralelismo* y *Buscar secuencia oculta*.

El algoritmo *Buscar secuencia* tiene como objetivo determinar si el proceso analizado se puede descomponer en otros subprocesos ordenados secuencialmente. En caso de ser posible la descomposición, se devuelve la lista de los bloques de construcción detectados, en caso contrario, se devuelve la lista vacía. Los subprocesos ordenados secuencialmente pueden ser claramente identificados debido a que estos están separados por una o varias actividades consecutivas que aparecen ocupando una columna completa en cada caso. En ocasiones este tipo de actividades no se puede identificar debido a que no se almacenaron en el registro de evento.

El pseudocódigo correspondiente al algoritmo se muestra en el Anexo 2.

El algoritmo *Buscar secuencia* se compone de dos grandes partes. La primera asociada con la búsqueda de las columnas que permiten separar dos bloques de construcción ordenados secuencialmente y la segunda parte está asociada a la conformación de los nuevos bloques de construcción. Esta segunda parte tiene asociada una complejidad temporal O($m \cdot n$), dado que es necesario recorrer todo el bloque de construcción analizado. Mientras que para la primera

parte es necesario recorrer todo el bloque de construcción y para cada columna que sea completa se verifica que sea un separador. Esta verificación se realiza usando una lista con las actividades que contiene cada columna del bloque de construcción, cuya dimensión es *m*. En correspondencia con lo antes analizado la complejidad temporal del algoritmo *Buscar secuencia* es O($m^2 \cdot n$).

El algoritmo *Buscar lazo* tiene como objetivo determinar si el proceso analizado se puede descomponer en un subproceso que se realiza múltiples veces. En caso de ser posible la descomposición, se devuelve una lista con un solo bloque de construcción, el cual, representa al subproceso que se repite; en el caso contrario, se devuelve la lista vacía. Para determinar un bloque de construcción que representa un subproceso que se repite en múltiples ocasiones es necesario identificar la actividad inicial de ese subproceso. Esta actividad inicial permitirá separar secuencias de actividades que posteriormente conforman las filas del nuevo bloque de construcción. Las secuencias repetidas se desechan.

El pseudocódigo correspondiente al algoritmo se muestra en el Anexo 3.

En el algoritmo *Buscar lazo* las operaciones que tienen mayor complejidad temporal son las correspondientes al paso cinco y seis. La primera de estas operaciones requiere que se recorra todo el bloque de construcción por lo cual su complejidad es O($m \cdot n$). Mientras que la alineación de las secuencias definidas (paso seis) tiene una complejidad temporal de O($l \cdot n + l^2$), siendo *l* el promedio de las longitudes de las secuencias a alinear. Para este cálculo se toma como referencia el trabajo de Bose y Van der Aalst [84]. Considerando que no es posible establecer ninguna relación de igualdad o desigualdad entre las variables *m*, *n* y *l* se determina que la complejidad temporal del algoritmo *Buscar lazo* es O($m \cdot n + l^2$).

El algoritmo *Buscar XOR-OR* tiene como objetivo determinar si el proceso analizado se puede descomponer en otros subprocesos ordenados como opciones de una selección. En caso de ser posible la descomposición, se devuelve la lista de los bloques de construcción

detectados, en caso contrario, se devuelve la lista vacía. Inicialmente se busca identificar una descomposición según XOR. Para determinar los bloques de construcción que representan opciones de una selección (XOR), se construyen conjuntos disjuntos con las actividades que componen el bloque de construcción analizado. Inicialmente existe un conjunto que contiene las actividades que comparten una fila del bloque de construcción analizado, posteriormente se unen los conjuntos que se intersectan mediante alguna actividad. Si al finalizar este proceso queda más de un conjunto, entonces se construyen los bloques de construcción que representan cada una de las opciones.

Si solo queda un conjunto entonces se busca identificar una descomposición según OR. Para ello se determinan las secuencias bases. Una secuencia base es una fila del bloque de construcción que no está compuesta completamente por la unión de otras filas. Las secuencias que contienen actividades comunes se unen en un mismo conjunto. Si al finalizar este proceso queda más de un conjunto, entonces se construyen los bloques de construcción que representan cada una de las opciones.

El pseudocódigo correspondiente al algoritmo se muestra en el Anexo 4.

El algoritmo *Buscar XOR-OR* tiene dos partes fundamentales, una asociada a la detección de una selección XOR y otra asociada a la detección de una selección OR. La primera de estas operaciones tiene una complejidad O(n^2) y está determinada por un recorrido por todo par de filas del bloque de construcción. Las operaciones que se realizan durante el recorrido están asociadas a la estructura de datos Conjuntos Disjuntos, en la cual la operación unión tiene una complejidad temporal de O(log*n) [150]. En la segunda parte las operaciones de mayor complejidad temporal están asociadas con la búsqueda de las filas base. Para determinar que una fila es base es necesario recorrer los m elementos de las restantes n-1filas del bloque de construcción y determinar las coincidencias. La complejidad temporal de estas operaciones es O($n^2 \cdot m$). El resto de las operaciones siguientes tiene menor complejidad temporal dado que

consiste en un recorrido por cada par de filas base (como máximo *n*) y la posible unión de estas (usando Conjuntos Disjuntos). A partir del análisis realizado se determina que la complejidad temporal del algoritmo *Buscar XOR-OR* es $O(n^2 \cdot m)$.

El algoritmo *Buscar paralelismo* tiene como objetivo determinar si el proceso analizado se puede descomponer en otros subprocesos ordenados en paralelo. En caso de ser posible la descomposición, se devuelve la lista de los bloques de construcción detectados, en caso contrario, se devuelve la lista vacía. Para determinar los bloques de construcción que representan subprocesos en paralelo se construyen conjuntos disjuntos con las actividades que componen el bloque de construcción analizado. Las actividades que pertenecen a conjuntos diferentes se encuentran en paralelo, mientras que las actividades que forman un mismo conjunto se relacionan mediante otro tipo de patrón de flujo de trabajo. Si como resultado se obtiene más de un conjunto, entonces los bloques de construcción que se forman a partir de estos representan subprocesos en paralelo.

El pseudocódigo correspondiente al algoritmo se muestra en el Anexo 5.

En el algoritmo *Buscar paralelismo* el paso de mayor complejidad temporal es el encargado de formar *k* grupos siguiendo un grupo de condiciones. Para formar los grupos de actividades es necesario que se tome como referencia cada actividad que conforma el bloque de construcción, lo que significa recorrer una lista de *m* elementos. Para cada actividad se verifica la forma en la que se relaciona con el resto de las actividades y para esto se recorre todo el bloque de construcción. En correspondencia con el análisis realizado la complejidad temporal del algoritmo *Buscar paralelismo* es $O(m^2 \cdot n)$.

El algoritmo *Buscar secuencia oculta* tiene como objetivo determinar si el proceso analizado se puede descomponer en otros subprocesos ordenados secuencialmente. La diferencia de este algoritmo con el algoritmo *Buscar secuencia* es que en este caso no existe una columna que no contiene símbolos vacíos (“–”) y que permite delimitar el final de un bloque de

construcción e inicio del bloque de construcción que le sigue. En consecuencia, se determinan las posibles soluciones (variantes de descomposición que representan subprocesos ordenados secuencialmente) considerando los aspectos que se enuncian a continuación.

- Cada bloque de construcción que conforma una solución se puede descomponer mediante XOR, OR, paralelismo o lazo.
- Las soluciones se evalúan y se seleccionan las mejores, considerando en la evaluación que los bloques de construcción formados disminuyen la cantidad de lazos y paralelismos rotos. Un lazo roto se evidencia cuando una actividad aparece múltiples veces en una misma fila en el bloque de construcción analizado y en la solución propuesta no aparece como parte de un único bloque de construcción. De manera análoga se determina la cantidad de paralelismos rotos.

El pseudocódigo correspondiente al algoritmo se muestra en el Anexo 6.

Para obtener la mejor solución para cada bloque de construcción se cuantifican los lazos y paralelismos rotos. Esto se hace mediante el valor de μ, el cual se calcula según la ecuación 1.

$$\mu = |D(i, j, BC)| + |P(i, j, BC)| \ , \ i, j \in \mathrm{N} \qquad \textbf{(1)}$$

$D(i, j, BC)$ es el conjunto de actividades que cumplen con la siguiente condición: la actividad es duplicada en la matriz contenida en el bloque de construcción *BC* y la actividad no es duplicada en el bloque que contiene la sub-matriz formada desde la columna *i* hasta la *j* de la matriz contenida en *BC*.

$P(i, j, BC)$ es el conjunto que contiene todas las relaciones de paralelismo que existían entre dos actividades en el bloque de construcción *BC* y no existen en el bloque que contiene la sub-matriz formada desde la columna *i* hasta la *j* de la matriz contenida en *BC*.

En el algoritmo *Buscar secuencia oculta* las operaciones de mayor complejidad temporal son las relacionadas con la formación y evaluación de cada posible bloque de construcción hijo (Pasos del 6 al 11 en el pseudocódigo del Anexo 6). Para cada intervalo de columnas del bloque de construcción original [*i*, *j*] se evalúa la mejor descomposición mediante el cálculo de μ. Este cálculo requiere de dos operaciones. La primera de estas está dirigida a contar la cantidad de tareas duplicadas en todo el bloque de construcción y dentro del bloque de construcción formado desde *i* hasta *j*. Esta operación tiene una complejidad temporal de $O(m^2 \cdot n)$, debido a que para cada actividad contenida en el bloque de construcción se recorre todo el bloque de construcción buscando repeticiones de dicha actividad. La segunda operación está dirigida a contabilizar la cantidad de pares de actividades que se encuentran en paralelo en todo el bloque de construcción y dentro del bloque de construcción formado desde *i* hasta *j*. La complejidad temporal de esta operación es $O(m^2 \cdot n)$, debido a que para cada actividad se determinan los pares de actividades en paralelo que él conforma. Considerando el análisis realizado la complejidad temporal del algoritmo *Buscar secuencia oculta* es $O(m^4 \cdot n)$.

2.8 Componente para la aplicación de los operadores de ausencia de información

Este componente tiene como propósito identificar las manifestaciones de ausencia de información y realizar la estimacion en correspondencia.

Para cada bloque de construcción identificado en el paso anterior se aplican un conjunto de operadores definidos para el manejo de la información ausente. En dependencia de las características del bloque de construcción puede no aplicarse la totalidad de los operadores propuestos. Estos son: *Operador de salto*, *Operador de lazo*, *Operador de división/unión*, *Operador de secuencia oculta* y *Operador probabilístico*.

Cada operador se aplica sobre el bloque de construcción que puede o no contener actividades invisibles estimadas por operadores antes empleados. El operador aplicado modifica el bloque de construcción sólo si detecta alguna actividad invisible.

Para analizar cada bloque de construcción se recorre el árbol de bloques de construcción *E* in-orden, al visitar un nodo del árbol (contiene un bloque de construcción) se aplican todos los operadores. Para reflejar la información estimada se crea un nuevo árbol de bloques de construcción estimados Ĕ.

Cada uno de los operadores enunciados y los algoritmos que permiten su aplicación fueron propuestos por el autor de la presente investigación [49, 50].

A continuación se describen cada uno de los operadores de estimación de información ausente propuestos.

2.8.1 Operador de salto

Este operador se aplica para identificar la *Situación de salto* (sección 1.2) antes descrita y estimar la información ausente correspondiente. Los pasos que se siguen están descritos en el Algoritmo 2.

Algoritmo 2 Operador de salto

Entrada: Bloque de construcción: *Ce*

Salida: Bloque de construcción

1: **Si** *Ce* es resultado de una descomposición según la relación OR o XOR **Entonces**

2: **Si** *Ce* contiene una sola fila y esta solo tiene símbolos vacíos (“–”) **Entonces**

3: Generar una actividad invisible λ_i y se adiciona a la primera columna de la matriz contenida en *Ce*. *i* se calcula en función de la cantidad de actividades invisibles estimadas para el proceso analizado

4: Devolver *Ce* modificado

FinSi

FinSi

En el algoritmo *Operador de salto* la operación de mayor complejidad temporal está en el paso 2, específicamente al verificarse si la fila está compuesta por símbolos vacíos. La longitud máxima de fila es *m*. En consecuencia con el análisis realizado la complejidad temporal del algoritmo *Operador de salto* es O(*m*).

2.8.2 Operador de lazo

Este operador se aplica para identificar situaciones en las que existen tareas duplicadas, específicamente formando un lazo de longitud uno (ver sección 1.2), y estimar la información ausente. Los pasos que se siguen están descritos en el Algoritmo 3.

Algoritmo 3 Operador de lazo

Entrada: Bloque de construcción: *Ce*

Salida: Bloque de construcción

1: **Si** *Ce* es resultado de una descomposición según la relación Lazo **Entonces**

2: **Si** *Ce* contiene una sola actividad **Entonces**

3: Generar una actividad invisible λ_i y se adiciona en una nueva columna que se coloca detrás de la actividad existente en *Ce*. *i* se calcula en función de la cantidad de actividades invisibles estimadas para el proceso analizado

4: Devolver *Ce* modificado

FinSi

FinSi

La complejidad temporal del algoritmo *Operador de lazo* es O(1).

2.8.3 Operador de división/unión

Este operador se aplica para identificar la *Situación de división/unión* (sección 1.2) antes descrita y estimar la información ausente. Los pasos que se siguen están descritos en el Algoritmo 4.

Algoritmo 4 Operador de división/unión

Entrada: Bloque de construcción: *Ce*, Lista de bloques de construcción obtenidos al descomponer a *Ce*: *List*

Salida: Bloque de construcción

1: **Si** *Ce* es resultado de una descomposición según la relación OR, XOR o paralelismo **Entonces**

2: **Si** |*List*| > 0 y *List*[1] se descompuso según la relación OR, XOR o paralelismo **Entonces**

3: **Si** la primera columna de *Ce* contiene símbolos vacíos **Entonces**

4: Insertar una nueva columna en la primera posición de *Ce* y en cada fila de la columna insertada se pone la misma actividad invisible λ_i. *i* se calcula en función de la cantidad de actividades invisibles estimadas para el proceso analizado

FinSi

5: **Si** la última columna de *Ce* contiene símbolos vacíos **Entonces**

6: Insertar una nueva columna en la última posición de *Ce* y en cada fila de la columna insertada se pone la misma actividad invisible λ_j, tal que $j \neq i$

FinSi

7: Devolver *Ce* modificado

FinSi

FinSi

En el algoritmo *Operador de lazo* las operaciones de mayor complejidad temporal son las asociadas a las verificaciones que detectan la aparición de símbolos vacíos en la primera y última columna y a la inserción de una nueva columna. En ambos casos la complejidad temporal es O(*n*) debido a que es necesario recorrer o crear *n* elementos. En consecuencia la complejidad temporal del algoritmo *Operador de lazo* es O(*n*).

2.8.4 Operador de secuencia oculta

Este operador se aplica para identificar la *Situación de secuencia oculta* (sección 1.2) antes descrita y estimar la información ausente. Los pasos que se siguen están descritos en el Algoritmo 5.

Algoritmo 5 Operador de secuencia oculta

Entrada: Bloque de construcción: *Ce*, árbol de bloques de construcción *E*

Salida: Lista de bloques de construcción

1: Crear una lista *List* y almacenar los bloques de construcción obtenidos al descomponer a *Ce*.

1: **Si** |*List*| > 0 y *List*[1] se descompuso según la relación *Secuencia oculta* **Entonces**

2: Crear *k* actividades invisibles. *k* = |*List*| -1

3: **Para** *i* = 1 **Hasta** *k* **Con Paso** 1 **Hacer**

4: Insertar una nueva columna en la última posición del bloque de construcción *List*[*i*] y en cada fila de la columna insertada se pone la misma actividad invisible λ_i

5: Insertar una nueva columna en la primera posición del bloque de construcción *List*[*i*+1] y en cada fila de la columna insertada se pone la misma actividad invisible λ_i

FinPara

6: Devolver *List* modificada

FinSi

La complejidad temporal del algoritmo Operador de secuencia oculta es O(*k*·*n*), donde *k* = (cantidad de bloques de construcción contenidos en *List*) -1. La complejidad está determinada considerando que deben crearse *k* nuevas columnas las cuales tienen *n* elementos, siendo *n* la cantidad de filas de *Ce*.

2.8.5 Operador probabilístico

Este operador se aplica para identificar la situación de *Opciones equiprobables* (sección 1.2) antes descrita y estimar la información ausente. Los pasos que se siguen están descritos en el Algoritmo 6.

Algoritmo 6 Operador probabilístico

Entrada: Bloque de construcción: *Ce*, Lista de bloques de construcción obtenidos al descomponer a *Ce*: *List*, árbol de bloques de construcción *E*

Salida: nuevo árbol de bloques de construcción Ĕ

1: **Si** |*List*| > 0 y *List*[1] se descompuso según la relación OR o XOR **Entonces**

2: **Para** cada bloque de construcción contenido en *List* **Hacer**

3: Calcular la cantidad τ de trazas que forman el bloque de construcción *Ce* según la ecuación 2

$$\tau = \sum_{j=1}^{n} \gamma_j \quad \textbf{(2)}$$

γ_j es la multiplicidad de la traza *j* y *n* la cantidad de filas del bloque de construcción analizado

4: Calcular la frecuencia relativa *fr* asociada al bloque de construcción según la ecuación 3

$$fr = \frac{\tau}{\sum_{i=1}^{|List|} \tau_i}, \quad 0 > fr < 1 \quad \textbf{(3)}$$

FinPara

5: Almacenar en *Lf* ordenadas de mayor a menor, todas las frecuencias relativas calculadas.

6: Crear una lista que se denominará *Menores* en la que se guarda el árbol que representa el caso base (caso en el que no se consideran actividades invisibles) y el correspondiente error. El error asociado a un árbol de manera general se calcula según la ecuación 4

$$E = \frac{\sum_{i=1}^{|List|} e_i}{|List|} \quad \textbf{(4)}$$

, e_i es el error asociado a cada opción de selección ubicada en un nodo. $e_i = |PN_j - fr_i|$, donde PN_j es la probabilidad de cada nodo hijo del nodo *j* y se calcula según las ecuación 5

$$PN_i = \frac{Probabilidad\ de\ ocurrencia\ del\ nodo\ padre\ (j)}{cantidad\ de\ hijos\ del\ nodo\ j} \quad \textbf{(5)}$$

7: **Para** *i* = 2 **Hasta** |*List*| **Con Paso** 1 **Hacer**

8: *Comb = SubArboles(Lf, i, probpadre, Ep, ε). Siendo probpadre* la probabilidad de ocurrencia del nodo padre y se inicializa en 1, *Ep* el error parcial que se va incrementando en la medida que se determina e_r para cada opción ubicada como nodo hoja en el árbol y se inicializa en 0, *ε es el e*rror asociado a *Menores*[1]. El procedimiento *SubArboles* permite determinar los subárboles en los cuales se consideran como nodos intermedios bloques de construcción que contienen actividades invisibles y como nodos hojas se tienen los bloques de construcción contenidos en *List*. La variable *i* condiciona la cantidad de subárboles que se generan. En la medida en que a cada árbol se le adiciona un nodo hoja se incrementa el *Ep* y se verifica que este no rebase el valor de *ε*. Si

se obtiene una solución con un error menor que ε, este nuevo error se asigna a ε. Esto garantiza que se limite el espacio de búsqueda de las soluciones.

9: **Si** |*Comb*| > 0 **Entonces**.

10: Adicionar a *Menores* el resultado de concatenar la raíz con las combinaciones contenidas en *Comb*.

FinSi

FinPara

11: Determinar la mejor solución considerando el menor *Ep* acumulado en cada elemento de *Menores*

12: **Si** la mejor solución es diferente de *Menores*[1] **Entonces**

13: Crear un nuevo árbol Ĕ según la estructura de la mejor solución encontrada, este nuevo árbol contiene la información estimada.

FinSi

14: Devolver Ĕ

FinSi

En el algoritmo *Operador probabilístico* las operaciones de mayor complejidad temporal están asociadas a los pasos siete y ocho, los cuales están asociados a la generación de todos los posibles árboles. Para el cálculo de la complejidad temporal de estas operaciones es necesario considerar que en el peor de los casos es necesario generar $w^2 \cdot 2^w$ estados, donde w es la cantidad de opciones en una selección. Un estado representa un nodo contenido en los árboles generados y se denota como *Estado*(i, j, k), tal que tal que i, j, $k \in$ N. La representación del estado se interpreta como que, de la lista *List* se toman las opciones desde la posición i hasta la posición j y para cada uno de estos w^2 elementos se consideran v probabilidades. La cantidad v está determinada por k, dado que cada probabilidad se calcula

como $p = {}^{1}/_{k}$, y el mayor valor que puede tomar k es 2^{w-1} considerando que la mayor cantidad de niveles que puede tener un árbol con w hojas es 2^{w-1}. La complejidad temporal de obtener cada estado es O(1). Un estado se obtiene a partir de los estados que se pueden formar con las opciones desde i hasta j contenidas en *List*, para esto es necesario considerar que obtener un estado asociado a un nodo hoja tiene una complejidad temporal de O(1). Para este último se calcula el error asociado al nodo hoja ($e_i = |PN_j - fr_i|$, fr_i se obtiene de List y PN_j se obtiene a partir de k). Considerando lo analizado anteriormente se concluye que la complejidad temporal del algoritmo *Operador probabilístico* es $O(w^2 \cdot 2^w)$. Hay que señalar que para evitar que se generen la totalidad de las soluciones se utiliza *Ep*, lo que permite desechar en cada momento las soluciones parciales que presentan un error superior a ε. Además, la cantidad de opciones de una selección en entornos reales difícilmente alcanzará valores superiores a diez.

2.8.6 Propagación de la información estimada

El bloque de construcción contenido en la raíz del árbol de bloques de construcción estimado Ĕ debe reflejar la totalidad de las actividades invisibles, por lo cual, se propaga la información estimada desde las hojas del árbol hasta la raíz. Cada bloque de contrucción contenido en Ĕ puede contener actividades invisibles, por lo que, para la propagación, se considera en cada caso la posición relativa de cada actividad en el bloque de construcción padre y tipo de descomposición que dio lugar al bloque de construcción analizado.

En este punto se valora el resultado de la estimación de información ausente realizada. Puede considerarse aplicar nuevamente el *Componente para la aplicación de los operadores de ausencia de información* considerando desechar o adicionar algún operador propuesto. Tambien es posible, antes de construir el registro de evento con la información estimada, considerar alguna modificación en el registro de evento original e iniciar nuevamente la estimación desde la aplicación del primero de los componentes propuestos en el modelo.

El pseudocódigo correspondiente al algoritmo Propagación de la información estimada se muestra en el Anexo 7. Este algoritmo fue propuesto por el autor del presente trabajo [49-50].

Para calcular la complejidad temporal del algoritmo (Anexo 7) se considera que la cantidad máxima de actividades estimadas es *m*, las cuales pueden ocupar *n* filas como máximo. En dependencia con la descomposición obtenida la cantidad máxima de bloques de construcción de un árbol es *m+m-1* o *n+n-1*. Para insertar una nueva actividad estimada en un bloque de construcción padre es necesario como máximo desplazar *m* columnas o afectar *n* filas en correspondencia con la descomposición realizada. Considerando el análisis antes realizado se concluye que la complejidad temporal del algoritmo es $O(n \cdot m \cdot (m+n)^2)$.

2.9 Componente para construir el registro de evento con información estimada

Este componente tiene como objetivo construir el registro de evento que contiene la información estimada.

Con tal propósito se eliminan del bloque de construcción contenido en la raíz del árbol Ĕ todos los símbolos vacíos ("–"). Sin considerar las actividades invisibles insertadas se buscan las coincidencias de cada caso contenido en el bloque de construcción (cada caso ocupa una fila de la matriz que constituye el bloque de construcción analizado) con los casos contenidos en el registro de evento original. Según las coincidencias se insertan las actividades invisibles en el registro de evento. El registro de evento modificado constituye la salida del modelo propuesto.

El pseudocódigo correspondiente al algoritmo *Construir registro de evento con IE* se muestra en el Anexo 8. Este algoritmo fue propuesto por el autor del presente trabajo en [49, 50].

Para el cálculo de la complejidad temporal del algoritmo propuesto es necesario considerar que se buscan en el registro de evento original los *n* casos del bloque de construcción contenido en la raíz del árbol Ĕ. El registro de evento original posee *k* casos, tal que $k \geq n$. La

cantidad máxima de actividades contenidas en un caso es m. En consecuencia con lo antes analizado la complejidad temporal del algoritmo *Construir registro de evento con IE* es O($k \cdot n \cdot m$).

2.10 Conclusiones del capítulo

En este capítulo se le dio cumplimiento a los objetivos específicos uno y dos planteados para la investigación.

Inicialmente se describen dos nuevas manifestaciones de ausencia de información (*Secuencia oculta de subprocesos* y *Opciones equiprobables*) que no se habían reportado en la literatura. Sus descripciones permiten hacer un análisis más completo del tratamiento de la ausencia de información.

El *Componente para determinar el árbol de bloques de construcción* permite la descomposición del proceso analizado. La descomposición se realizó considerando los patrones básicos de flujo de trabajo, Secuencia, Lazo, Selección (OR o XOR) y Paralelismo. Estos patrones se identificaron a nivel de subprocesos lo cual constituye un aporte de este trabajo. La descomposición permite representar como un bloque de construcción un determinado proceso. Esto constituye la base necesaria para posteriormente identificar las manifestaciones de ausencia de información.

Mediante un conjunto de operadores desarrollados se detecta y estima la información ausente en el registro de evento. Para la estimación de información ausente pueden utilizarse la totalidad de los operadores propuestos o no, así, en dependencia del contexto y del registro de evento analizado se puede flexibilizar la aplicación del modelo propuesto.

La estimación se realizó mediante algoritmos que presentan una complejidad temporal polinomial. Esto constituye un resultado importante dado que la cantidad de casos de un registro de evento pueden estar en el orden de los cientos a los miles. Solo en el caso del

Operador probabilístico la complejidad temporal no se comporta de esta forma, pero este operador se aplica en situaciones específicas (opciones equiprobables) y sobre un solo bloque de construcción. Se ha empleado una cota de error que permite reducir el espacio de búsqueda en la medida en que se va formando una posible solución, además, las opciones en las cuales se descompone el subproceso en entornos reales no tienden a sobrepasar las diez.

Es necesario resaltar que el modelo desarrollado para la estimación de información ausente, al aplicarse en la etapa de pre-procesamiento, resulta novedoso en la forma en la que trata la problemática abordada.

CAPÍTULO 3. ANÁLISIS DE RESULTADOS

En este capítulo se presenta la aplicación desarrollada a partir del modelo EIAT. Haciendo uso de la aplicación informática desarrollada se evalúan un conjunto de procesos y se analizan los resultados obtenidos.

3.1 Aplicación informática para la estimación de información ausente.

Se desarrolló una aplicación informática que permite realizar la estimación de información ausente a partir de las trazas usadas en la minería de proceso. La herramienta posibilita el uso de un registro de evento en el formato XES o MXML como entrada. Como salida se genera un fichero con la información estimada y en formato XES o MXML según corresponda. El desarrollo se realizó utilizando Java como lenguaje de codificación.

La aplicación desarrollada posibilita transitar por tres momentos importantes en la estimación de información ausente. En un primer momento la aplicación permite leer un registro de evento y configurar los parámetros necesarios para la alineación de las trazas. Para el desarrollo del componente de alineación de las trazas se utilizó el plugin desarrollado por Bose y Van der Aalst [41]. No se utiliza la última versión implementada de la técnica Trace Alignment debido a que en esta se separa la agrupación de las trazas de la alineación. Esto no influye en la aplicación de la herramienta debido a que en las pruebas realizadas se crea para cada proceso un solo grupo de trazas. En la Figura 3.1 se muestra una imagen de la aplicación en la que se muestra la alineación de un registro de evento. En la parte derecha de la Figura 3.1 aparece el panel correspondiente a los parámetros para la configuración de la alineación.

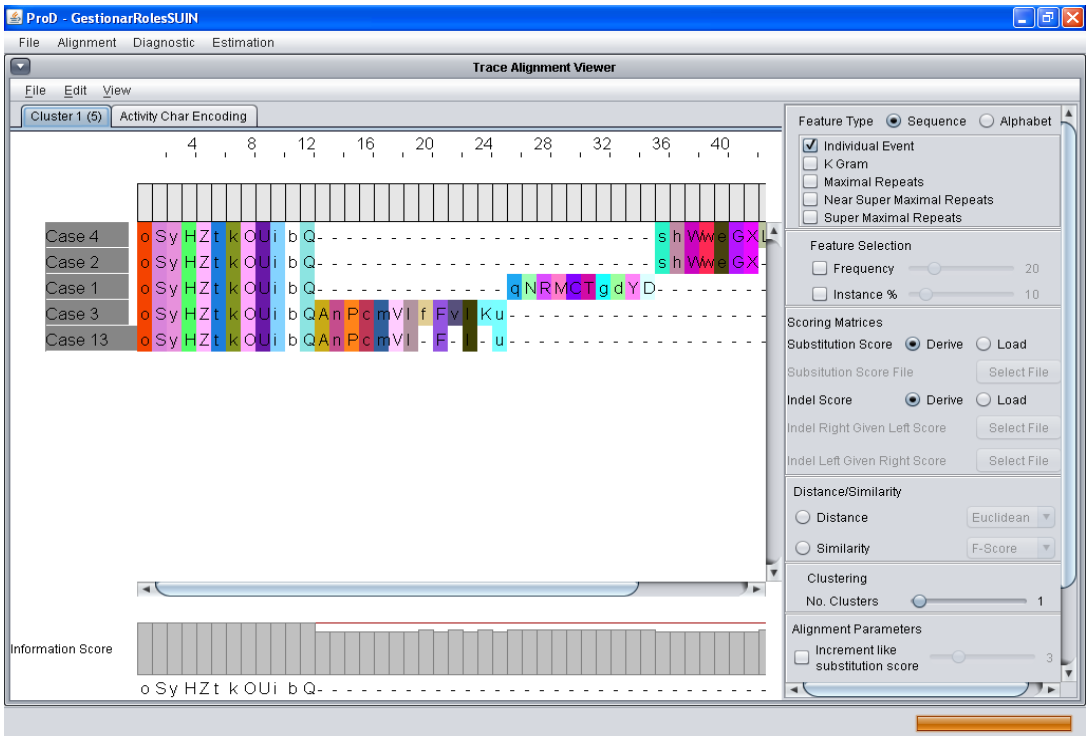

Figura 3.1. Alineación de las trazas.

Posterior a la alineación de las trazas se realiza la descomposición del proceso. La Figura 3.2 muestra el árbol de bloques de construcción producto de la descomposición del proceso analizado. En la figura mencionada se muestran tres paneles verticales, el primero (de izquierda a derecha) muestra las agrupaciones creadas en la alineación. A partir de seleccionar una agrupación se muestra en el panel del centro el árbol de bloques de construcción producto de su descomposición. En cada nodo del árbol se muestra el nombre del bloque de construcción creado y las relaciones que se establecen entre los bloques de construcción se reflejan en los arcos de dos formas, mediante un mensaje de texto y utilizando colores diferentes para cada tipo de relación. Los nodos de color verde representan hojas del árbol, los azules bloques de construcción que no pudieron descomponerse y los negros nodos intermedios. El árbol se puede expandir o contraer según se desee.

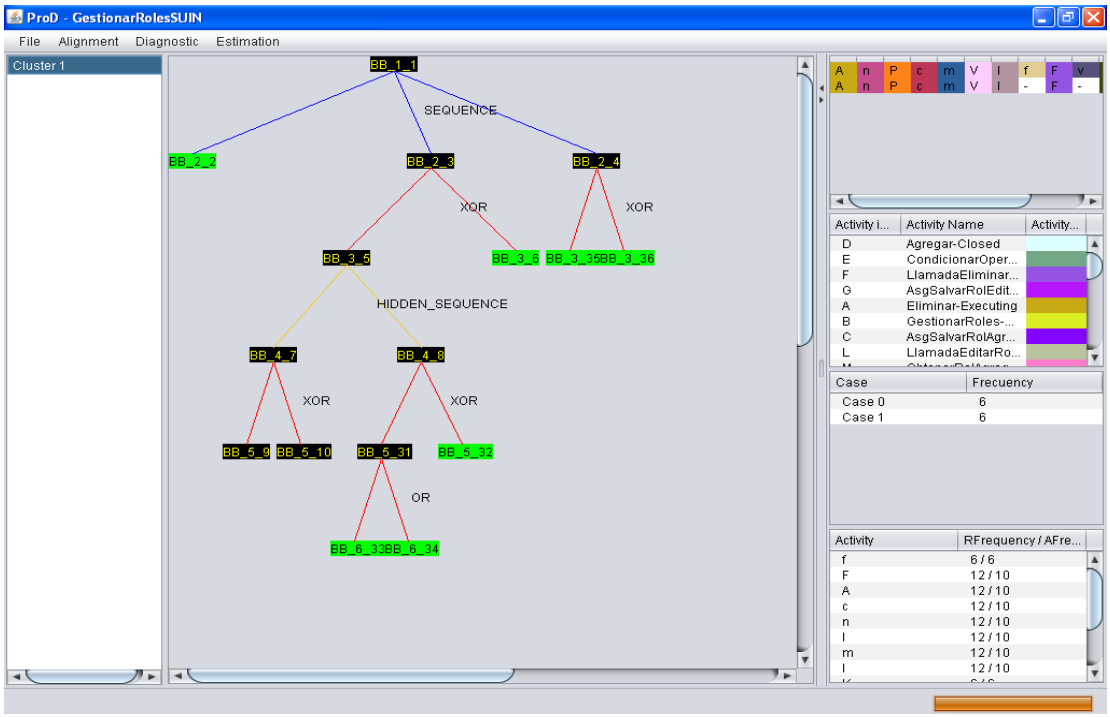

Figura 3.2. Árbol de bloques de construcción.

En el panel derecho se muestran cuatro secciones. La primera (de arriba hacia abajo) muestra el contenido del bloque de construcción selecionado en el panel del centro. En la segunda sección se muestra la leyenda construida durante la alineación de las trazas. En la siguiente sección se muestra la frecuencia de ocurrencia de los casos que conforman el bloque de construcción seleccionado. En la última sección se muestra la frecuencia relativa de ocurrencia de las actividades que conforman el bloque de contrucción seleccionado.

En un último momento se realiza la estimación de información ausente a partir del árbol de bloques de construcción obtenido anteriormente.

La Figura 3.3 muestra la estimación realizada a un proceso analizado. En ella se muestran tres paneles verticales. El primero (de izquierda a derecha) muestra en la sección superior el identificador del árbol asociado a cada grupo obtenido en la alineación. En la sección inferior aparecen los operadores que pueden aplicarse al bloque de construcción selecionado o todos los bloques de construcción que conforman el árbol obtenido anteriormente.

Figura 3.3. Estimación de información ausente.

En el panel del centro se muestra el árbol, de bloques de construcción obtenido, así como los detalles del bloque de construcción selecionado y la leyenda.

En el panel de la derecha se muestra en su sección superior el árbol de bloques de construcción estimado. En la segunda sección aparecen los detalles del bloque de construcción seleccionado en la sección anterior y en la última sección se muestra un resumen de las actividades invisibles que fueron insertadas y la causa por la cual se adicionó.

En la parte superior derecha de la Figura 3.3 aparece un menú que permite salvar la solución obtenida a partir de la estimación de información ausente.

3.2 Resultados experimentales.

Se definió un conjunto de experimentos para probar la efectividad del modelo EIAT empleándose para ello la aplicación informática expuesta anteriormente.

3.2.1 Diseño experimental

Para el experimento se definen cuatro grupos de procesos y tres momentos. Cada grupo está asociado a un proceso de negocio. Los momentos están asociados a las etapas por las que transita un proceso, es decir, la evaluación a partir de un registro de evento en su estado original, con ausencia de información y con información estimada. La Tabla 3.1 muestra el diseño experimental realizado y la simbología utilizada es la siguiente:

- G: Grupo de participantes. En este caso cada grupo está compuesto por 45 procesos, 15 asociados a cada momento (Original, Ausencia de Información e Información estimada).
- R: Asignación al alzar.
- X: Tratamiento o estímulo. En este caso X_1 se corresponde con la extracción de información del registro de evento analizado en cada grupo durante el primer momento. X_2 se corresponde con la aplicación del modelo propuesto para la estimación de información ausente.
- O: Observación.

Tabla 3.1. Diseño experimental propuesto.

	Original		Ausencia de información		Información estimada
R G_1	O_1	X_1	O_2	X_2	O_3
R G_2	O_4	X_1	O_5	X_2	O_6
R G_3	O_7	X_1	O_8	X_2	O_9
R G_4	O_{10}	X_1	O_{11}	X_2	O_{12}

Se han empleado cuatro procesos diferentes, uno para cada grupo. Se aplicaron dos algoritmos de descubrimiento, Alpha++ e ILP. A partir de su aplicación se evaluaron las cinco métricas seleccionadas en el capítulo uno sección 1.5 (Fitness Unsatisfied, Fitness

Unhandled, ETCPrecision, Non Fit Traces y Improved Structural Appropriateness), por lo cual se emplea el mismo diseño experimental propuesto para cada evaluación realizada y en correspondencia con la métrica y el algoritmo utilizado.

En el primer momento (Original) las observaciones O_1, O_4, O_7 y O_{10} representan la evaluación de la métrica analizada para el registro de evento en su estado original. En el segundo momento (Ausencia de información) las observaciones O_2, O_5, O_8 y O_{11} están asociadas a la evaluación de la métrica analizada después de extraer información del registro de evento original (X_1) y aplicar el algoritmo de descubrimiento. En este punto para cada observación se hacen 15 mediciones. Al registro de evento original se le extrajo de manera aleatoria el 3, 5 y 10 % de la información. Se formaron tres grupos de cinco procesos cada uno en correspondencia con los porcentajes de ausencia de información. Los registros de eventos que conforman un grupo son diferentes.

En el último momento (Información estimada) las observaciones O_3, O_6, O_9 y O_{12} están asociadas a la evaluación de la métrica analizada después de aplicar el modelo propuesto (X_2) y el algoritmo de descubrimiento. Cada registro de evento con ausencia de información se transformó usando la aplicación informática desarrollada y se obtuvo un nuevo registro de evento con la información estimada. En consecuencia, cada observación asociada a este momento contiene 15 registros de eventos.

Para demostrar la hipótesis planteada en la investigación se realizaron varias pruebas. La primera estuvo dirigida a comparar mediante un análisis de varianza de segunda vía no paramétrico de Friedman, los datos asociados a cada uno de los momentos enunciados para un grupo específico. Esta evaluación buscó detectar diferencias significativas entre las observaciones (por ejemplo O_1, O_2 y O_3). Los valores observados debían decrecer para el segundo de los momentos (Ausencia de información) y aumentar en el último (Información estimada).

Luego se realizaron comparaciones por pares en un grupo utilizando el test no paramétrico de signos con rangos de Wilcoxon. Se analizan los datos correspondientes al primer y segundo momento, por ejemplo O_1 y O_2, buscando detectar de manera significativa un predominio del decremento en la segunda observación. También se analizan los datos correspondientes al segundo y tercer momento, por ejemplo O_2 y O_3, buscando detectar de manera significativa un predominio del incremento en la tercera observación. Por último, se analizan los datos correspondientes al primer y tercer momento, por ejemplo O_1 y O_3, buscando detectar un empate entre los valores, lo cual no revelaría diferencias significativas.

Se realizan análisis transversales que permiten comparar las observaciones de los diferentes grupos en un mismo momento, por ejemplo O_1, O_4, O_7 y O_{10}. En estos casos, se espera detectar diferencias significativas al aplicar el test de Kruskal-Wallis. La diferencia entre los valores estaría determinada por el hecho de que los registros de eventos analizados presentan diferentes características, lo que influye en la complejidad del proceso de descubrimiento y en la evaluación de las métricas utilizadas.

Para entender las diferencias detectadas se realizan las comparaciones por pares utilizando el test de Mann-Whitney. Se espera detectar diferencias significativas entre las observaciones asociadas a los registros de eventos con diferentes características.

3.2.2 Características de los procesos analizados

Se utilizaron para realizar las pruebas cuatro procesos que fueron generados utilizando la aplicación informática Process Log Generator en la versión 1.4 beta [151]. Se decidió utilizar una herramienta que generara de manera aleatoria un registro de evento artificial, debido a que, en el área de minería de proceso no se detectó una base de datos que contenga procesos que puedan ser utilizados para la validación de las técnicas desarrolladas. Aun cuando existen algunos registros de eventos asociados a procesos reales estos no cuentan con las

características adecuadas para realizar una correcta validación de la propuesta. Los principales problemas están relacionados con el reflejo parcial de los patrones de flujo de trabajo y la cantidad de casos. Hay que señalar además que de estos registros de eventos no se conoce el proceso original a partir del cual tienen lugar, en consecuencia, no es posible saber qué actividades pueden estar faltando o cuánto ruido pudo manifestarse.

Un registro de evento generado de manera artificial puede reflejar los patrones de flujo de control conocidos, tener diferentes niveles de patrones anidados y la cantidad de casos puede variar en correspondencia con los elementos enunciados.

Las características de los registros de eventos generados son las siguientes:

Tabla 3.2. Descripción de los registros de evento.

Atributos	Proceso 1	Proceso 2	Proceso 3	Proceso 4
Cantidad de casos	100	1000	2000	500
Eventos	762	10400	23145	6655
Clases de eventos	18	34	28	40
Cantidad de patrones anidados	2	3	3	3
Refleja patrones de flujo de trabajo: Secuencia, AND Split/join, XOR Split/join	Si	Si	Si	Si
Refleja el patrón de flujo de trabajo Lazo	No	No	No	Si

Para la conformación de los registros de eventos se fueron incrementando las variables cantidad de casos, cantidad de patrones anidados y la probabilidad de reflejar los patrones de flujo de trabajo. La complejidad del descubrimiento aumenta en la medida en la que se manifiestan, en el registro de evento, mayor cantidad de variables y estas reflejan un aumento. Es necesario señalar que no es objetivo de este trabajo establecer una relación entre las características de un registro de evento y la complejidad en el descubrimiento del modelo de proceso representativo de este.

Hay que considerar que los algoritmos de descubrimiento de procesos se diferencian en la forma en la que manejan aspectos como los patrones de flujo de trabajo, por lo cual, variables

como la cantidad de casos influyen en el resultado pero no necesariamente determinan una mejor solución.

De los registros de eventos generados el que se considera de menor complejidad, para el descubrimiento, es el asociado al Proceso 1. El de mayor complejidad es el asociado al Proceso 4, debido a que, aun cuando no tiene la mayor cantidad de casos, si posee la mayor cantidad de clases de eventos y refleja todos los patrones de flujo de trabajo que se enuncian.

3.2.3 Algoritmos de descubrimiento utilizados

En la experimentación se utilizaron dos técnicas de descubrimiento de procesos, el Alpha++ [38, 109] y el ILP [40, 126]. Ambas técnicas tienen como resultado una Red de Workflow que representa el proceso descubierto. Esto da la posibilidad de que se muestren como recuadros grises las actividades invisibles que sean detectadas.

Según el análisis realizado, el algoritmo Alpha++ cubre tres de las ocho situaciones de ausencia de información enunciadas en el capítulo uno (Actividades invisibles contra Actividades duplicadas, Actividades invisibles contra lazos y Actividades invisibles contra sincronización). Para la aplicación de esta técnica se utiliza la implementación realizada en la herramienta ProM en su versión 6.1 [107].

El algoritmo ILP cubre cuatro de las ocho situaciones de ausencia de información enunciadas en el Capitulo 1 (Actividades invisibles contra Actividades duplicadas, Actividades invisibles contra lazos, Actividades invisibles contra sincronización y Lazos contra actividades invisibles junto a actividades duplicadas). Para la aplicación de esta técnica se utiliza la implementación realizada en la herramienta ProM en su versión 6.1 [107].

No se seleccionan otras técnicas analizadas que tienen mejor desempeño que los seleccionados, por ejemplo Genetic Miner, debido a que estas no satisfacen las situaciones de ausencia de información agregando al modelo descubierto una actividad invisible. El modelo

propuesto estima la información ausente a partir del registro de evento por lo cual se elimina la ambigüedad en la interpretación de este tipo de soluciones, independientemente del algoritmo de descubrimiento que se emplee.

3.2.4 Análisis de los resultados

Considerando el diseño experimental expuesto se realizan un conjunto de pruebas. La primera estuvo dirigida a comparar mediante un análisis de varianza de segunda vía no paramétrico de Friedman, los datos asociados a cada uno de los momentos enunciados para un grupo específico. Esta evaluación detectó diferencias altamente significativas (significación 0.000 < 0.01) entre las observaciones. Los valores observados disminuyeron para el segundo de los momentos y aumentaron en el último. La Figura 3.4 muestra los valores de los rangos medios para cada grupo al aplicar ambos algoritmos y medir las métricas Fitness Unsatisfied y Fitness Unhandled. Para ambas métricas se obtienen los mismos valores, por lo cual, se representan en una misma figura.

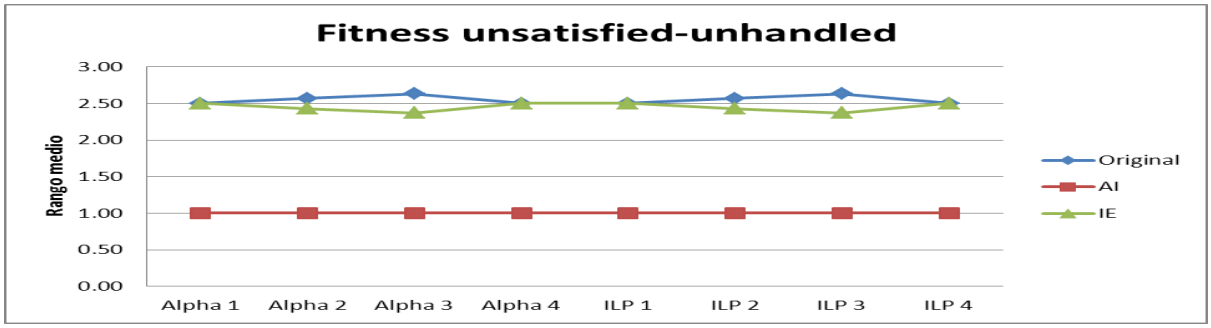

Figura 3.4. Evaluación de los rangos medios para los diferentes grupos, métrica Fitness Unsatisfied y Fitness Unhandled.

Las comparaciones por pares en un grupo se realizaron utilizando el test no paramétrico de signos con rangos de Wilcoxon. Se analizaron los datos correspondientes al primer y segundo momento y se detectó de manera altamente significativa (significación 0.000 < 0.01) un predominio del decremento en la segunda observación (AI, siglas de Ausencia de información). También se analizaron los datos correspondientes al segundo y tercer momento

y se detecta de manera altamente significativa (significación 0.000 < 0.01) un predominio del incremento en la tercera observación (IE, siglas de Información estimada). Por último, se analizaron los datos correspondientes al primer y tercer momento y se detecta un empate entre los valores, lo cual no revela diferencias significativas (significación 1.000).

Queda demostrado que la ausencia de información produce una reducción en esta medida que luego se recupera al estimar la información ausente utilizando el modelo propuesto.

Análogamente se realiza la evaluación para la métrica ETCPrecision. En la Figura 3.5 se muestra los valores de los rangos medios para cada grupo al aplicar ambos algoritmos y medir la métrica. El resultado de la evaluación es semejante a los obtenidos para las métricas Fitness Unsatisfied y Fitness Unhandled, excepto para la evaluación realizada al grupo 4 al aplicar el algoritmo ILP. Al hacer las comparaciones por pares en el grupo utilizando el test no paramétrico de signos con rangos de Wilcoxon no se aprecian diferencias significativas.

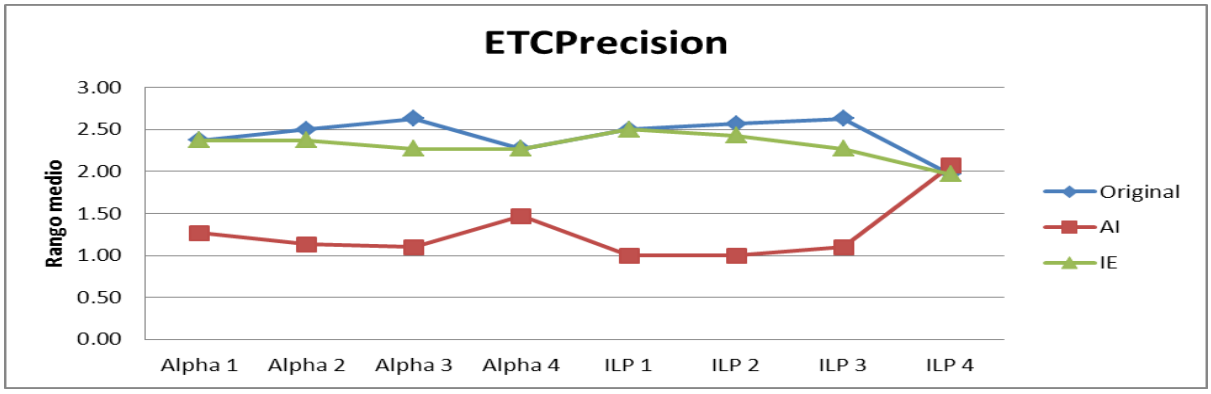

Figura 3.5. Evaluación de los rangos medios para los diferentes grupos, métrica ETCPrecision.

El grupo 4 se evalúa a partir del Proceso 4 generado. El registro de evento correspondiente presenta mayor cantidad de clases de eventos y refleja el patrón de flujo de trabajo Lazo. Esto propicia que los algoritmos de descubrimiento utilizados descubran modelos menos precisos. El valor de la métrica aumenta en el momento asociado a la Ausencia de información porque algunas de las actividades que se eliminaron del registro de evento original determinaban un lazo, por lo cual se restringe el comportamiento reflejado en el modelo descubierto. Mientras

que el modelo descubierto utilizando el registro de evento original y el que posee la información estimada genera modelos que generalizan el comportamiento asociado a cada lazo y por tanto disminuye la precisión.

Aun considerando los aspectos antes señalados para la comparación de las observaciones del primer y último momento predominan los empates, por lo cual, no hay diferencias significativas (significación 1.000). Esto demuestra que para la medida analizada la aplicación del modelo propuesto soluciona las afectaciones provocadas por la ausencia de información sobre el registro de evento original.

Se realiza la evaluación para la métrica Non Fit Traces de manera análoga a como se realizó para las métricas Fitness Unsatisfied y Fitness Unhandled. En la Figura 3.6 se muestran los valores de los rangos medios para cada grupo al aplicar ambos algoritmos y medir la métrica.

El resultado de la evaluación es semejante a los obtenidos para las métricas Fitness Unsatisfied y Fitness Unhandled, excepto para la evaluación realizada al grupo 4 aplicando el algoritmo de descubrimiento Alpha++. Este resultado se debe a que al aplicar el algoritmo Alpha++ sobre un registro de evento que refleja el patrón de flujo de trabajo Lazo se obtiene un modelo que generaliza el comportamiento observado, esto provoca que las trazas contenidas en el registro de evento no puedan reproducirse completamente sobre el modelo.

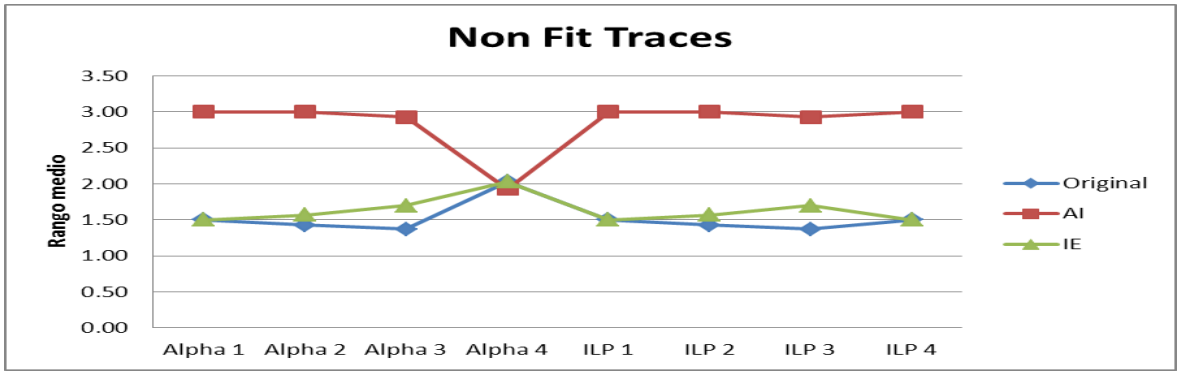

Figura 3.6. Evaluación de los rangos medios para los diferentes grupos, métrica Non Fit Traces.

La disminución de los valores asociados a la evaluación en el segundo momento se debe a que el modelo descubierto, a partir del registro de evento con ausencia de información, restringe el comportamiento observado en el registro de evento original, lo que propicia que mayor cantidad de trazas puedan reproducirse sobre el modelo.

Esta situación no se manifiesta en la evaluación de los modelos obtenidos utilizando el algoritmo ILP, debido a que, este busca en cada momento cubrir con el modelo obtenido el comportamiento observado en cada una de las trazas analizadas.

La comparación de las observaciones del primer y último momento revela un predominio de los empates, en correspondencia no existen diferencias significativas (significación 1.000). Se evidencia entonces que, para la medida analizada la aplicación del modelo propuesto soluciona las afectaciones provocadas por la ausencia de información sobre el registro de evento original.

La evaluación de la métrica Improved Structural Appropriateness (por sus siglas ISA) para todos los modelos obtenidos es 1.0. Esto revela que los algoritmos aplicados, independientemente del momento, no introducen en el modelo descubierto problemas estructurales (presencia de actividades invisibles redundantes y actividades duplicadas alternativamente), lo cual es positivo dado que los problemas estructurales dificultan el entendimiento del proceso analizado.

Se realizaron análisis transversales que permitieron comparar las observaciones de los diferentes grupos en un mismo momento considerando las diferentes métricas evaluadas.

La Figura 3.7 muestra un resumen de los resultados de los rangos medios de la medida de la métrica Fitness Unsatisfied. Se consideran los diferentes grupos en todos los momentos y utilizando los dos algoritmos de descubrimiento. En el eje de las abscisas de la Figura 3.7 se muestran los diferentes momentos asociados al algoritmo de descubrimiento utilizado.

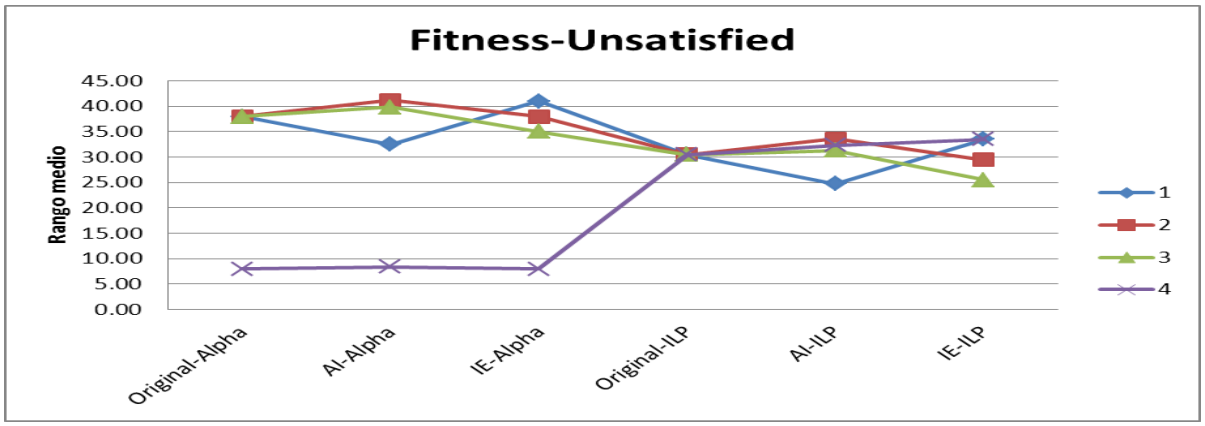

Figura 3.7. Evaluación de los rangos medios para los diferentes momentos, métrica Fitness Unsatisfied.

Se aplica el test de Kruskal-Wallis a las observaciones en un mismo momento y asociado a un algoritmo de descubrimiento. Para los tres momentos en el caso del algoritmo Alpha++ se detectan diferencias altamente significativas (significación 0.000 < 0.01).

Para una mayor comprensión de las diferencias detectadas se realizaron las comparaciones por pares utilizando el test de Mann-Whitney. Entre los grupos 1, 2 y 3 no aparecen diferencias significativas (significación > 0.01). Al comparar cada uno de los grupos 1, 2 y 3 con el grupo 4 aparecen diferencias significativas (significación 0.000 < 0.01). El grupo 4 está asociado al registro de evento que tiene mayor cantidad de clases de eventos y refleja el patrón de flujo de trabajo Lazo, lo cual determina un menor resultado en la evaluación de la métrica al aplicar el algoritmo Alpha++.

Para todos los tres momentos en el caso del algoritmo ILP no se detectan diferencias significativas (significación > 0.01).

La Figura 3.8 muestra un resumen de los resultados de los rangos medios de la medida de la métrica Fitness Unhandled. Se consideran los diferentes grupos en todos los momentos y utilizando los dos algoritmos de descubrimiento.

Se realizó un análisis semejante al que se hizo con la métrica Fitness Unsatisfied antes expuesta. Los resultados del análisis son análogos.

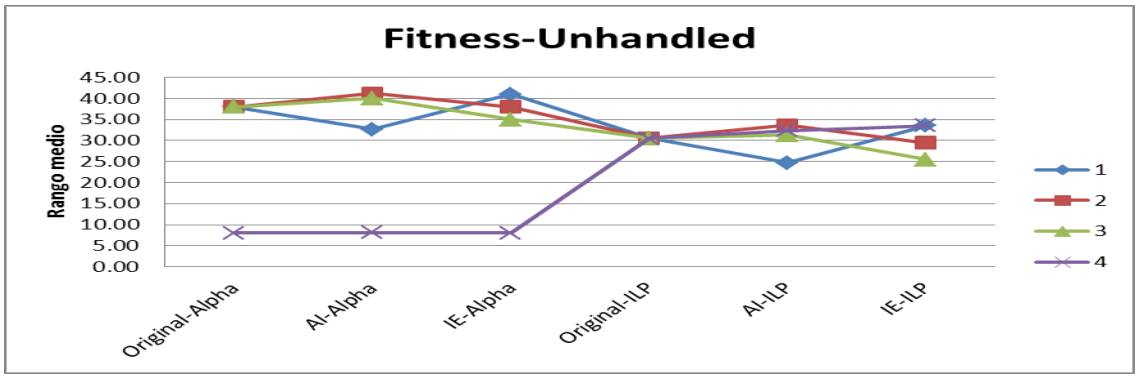

Figura 3.8. Evaluación de los rangos medios para los diferentes momentos, métrica Fitness Unhandled.

La Figura 3.9 muestra un resumen de los resultados de los rangos medios de la medida de la métrica ETCPrecision. Se consideran los diferentes grupos en todos los momentos y utilizando los dos algoritmos de descubrimiento.

Se aplica el test de Kruskal-Wallis a las observaciones en un mismo momento y asociado a un algoritmo de descubrimiento. Para los tres momentos y para los dos algoritmos utilizados se detectan diferencias significativas (significación 0.000 < 0.01) en las mediciones. Esta diferencia está determinada por las diferencias entre los registros de eventos generados, los cuales, difieren respecto a la cantidad de casos, cantidad de clases de eventos, cantidad de niveles de patrones anidados y los patrones de flujo de trabajo reflejados.

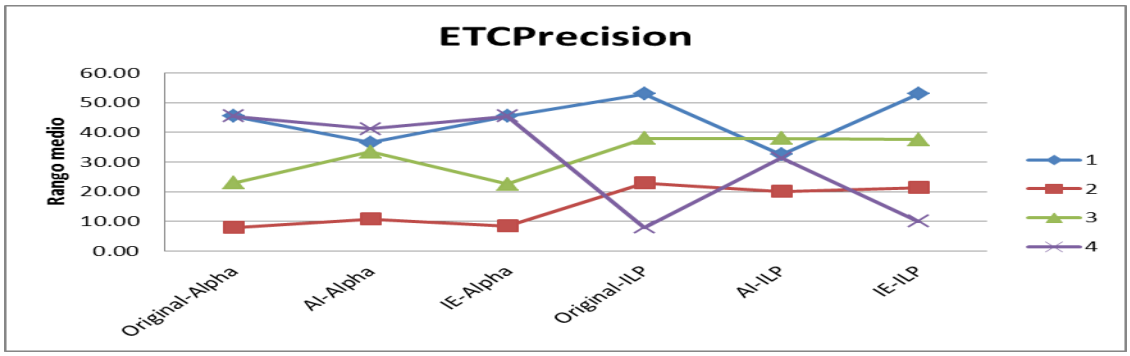

Figura 3.9. Evaluación de los rangos medios para los diferentes momentos, métrica ETCPrecision.

La Figura 3.10 muestra un resumen de los resultados de los rangos medios de la medida de la métrica Non Fit Traces. Se consideran los diferentes grupos en todos los momentos y utilizando los dos algoritmos de descubrimiento.

Se realizó un análisis semejante al que se hizo con la métrica Fitness Unsatisfied antes expuesta. Los resultados del análisis son análogos.

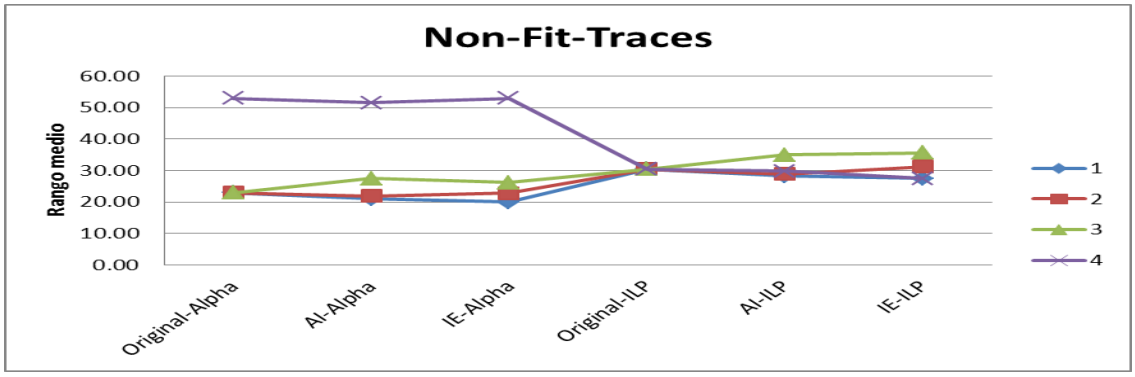

Figura 3.10. Evaluación de los rangos medios para los diferentes momentos, métrica Non Fit Traces.

Las pruebas realizadas haciendo uso de la aplicación informática desarrollada arrojaron que, aun cuando existen registros de eventos diferentes, la ausencia de información produce generalmente una reducción en las medidas de Fitness Unsatisfied, Fitness Unhandled, Precision y Non Fit Traces, que luego se recupera al estimar la información ausente utilizando el modelo propuesto. El modelo propuesto no introduce afectaciones estructurales, lo cual se evidencia en los resultados obtenidos para la métrica ISA. Esto confirma que la aplicación del modelo propuesto disminuye las afectaciones provocadas por la ausencia de información sobre la estructura y compresión del modelo descubierto.

3.3 Aplicación de la propuesta en un entorno cubano

Para la aplicación de la propuesta en un entorno cubano se realizó un análisis de un registro de evento almacenado por el Sistema Único de Identificación Nacional (SUIN), específicamente del módulo Gestionar Recursos. El SUIN fue desarrollado por el Ministerio del Interior de Cuba en conjunto con la Universidad de las Ciencias Informáticas.

El proceso Gestionar Recursos a partir del cual se generó el registro de evento utilizado tiene como objetivo gestionar las operaciones que se realizan sobre cualquier recurso del entorno informatizado. Entiéndase que un recurso puede ser desde un dispositivo de hardware hasta

una interfaz del SUIN. Los recursos se asignan a los roles creados en una determinada entidad.

El SUIN fue desarrollado utilizando la tecnología .Net, específicamente Windows Workflow Foundation [152]. Como base para la implementación del flujo de trabajo correspondiente al proceso Gestionar Recursos se utilizó el modelo que se muestra en el Anexo 9.

La selección de este proceso se realiza considerando dos aspectos fundamentales. El primero está asociado a la petición de la dirección del proyecto de analizar sus registros de eventos para identificar el proceso que se ejecuta en la práctica. En este sentido las técnicas de descubrimiento usadas en la minería de proceso brindan una abstracción del proceso ejecutado. Sin embargo, era necesario realizar una estimación de la información ausente para obtener un registro de evento que constituya una buena herramienta para la compresión del proceso analizado. El segundo de los aspectos que rige como motivación fue la facilidad para extraer el registro de eventos y la calidad del mismo. El proceso seleccionado está informatizado a partir de una definición explícita, lo cual denota un profundo conocimiento del proceso y constituye una ventaja para su control y actualización. En el SUIN existe un correcto mecanismo de registro de los casos o instancias del proceso lo cual facilita la extracción de un registro de eventos en el formato XES o MXML. La mayoría de los sistemas a los que tiene acceso el autor del presente trabajo no registran los eventos en correspondencia con la ejecución de cada instancia o caso del proceso. En consecuencia, extraer un registro de evento válido para el análisis requiere analizar la información registrada en las trazas y su posterior transformación, lo cual resulta en ocasiones complicado o imposible dada la poca información con que se cuenta.

El registro de evento utilizado presenta 51 casos, 11011 eventos, 90 clases de eventos, 3 tipos de eventos (Execute, Closed y Faulting) y 47 participantes.

Inicialmente haciendo uso de la herramienta informática desarrollada se alinean las trazas en un solo grupo. Posteriormente se realiza un análisis de las trazas alineadas y se determina que solo siete casos son completos. A pesar de esto, se decide trabajar con los 51 casos dado que de extraer los casos incompletos se perdería representatividad, es decir, el registro de evento representaría solo una parte del comportamiento registrado del proceso. Si se dejan solo los siete casos completos no se aprecian eventos relacionados con el subproceso Agregar recurso. Después de este análisis se descompone el proceso analizado. La Figura 3.11 muestra parcialmente el árbol de bloques de construcción obtenido.

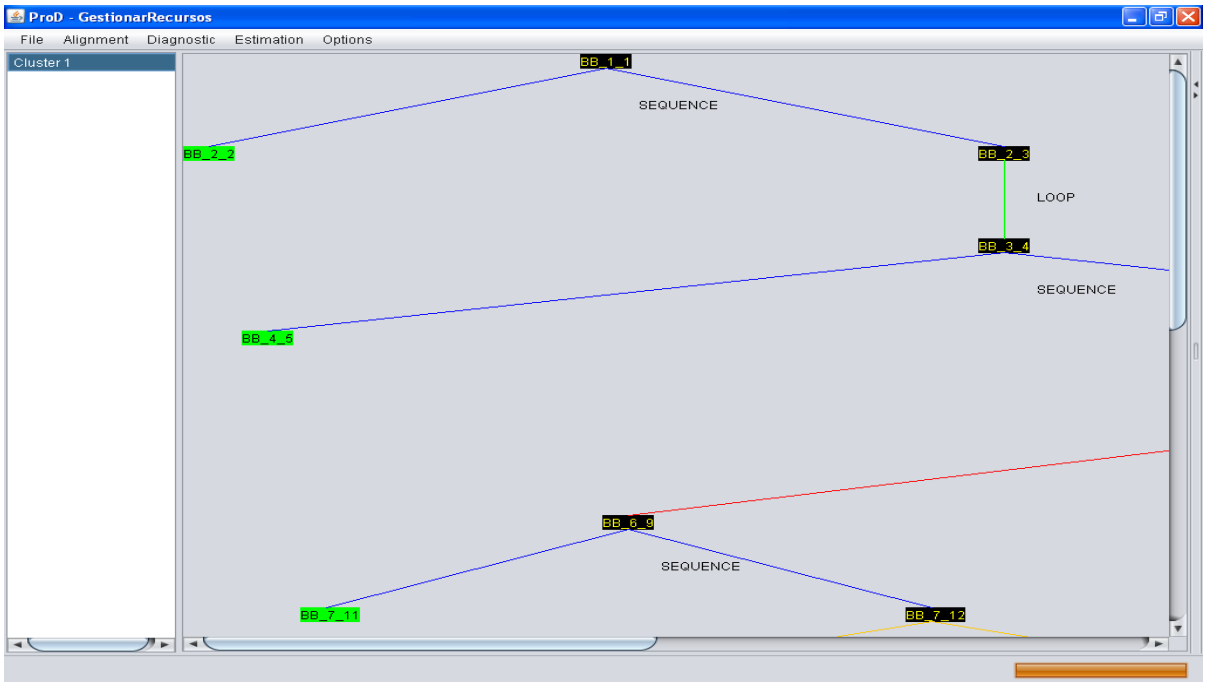

Figura 3.11. Detalle del árbol de bloques de construcción correspondiente al proceso Gestionar Recursos.

A partir de este árbol se realiza la estimación de información ausente y se obtienen ocho actividades invisibles. La Figura 3.12 muestra el resultado de la estimación de información ausente realizada para el proceso Gestionar Recursos.

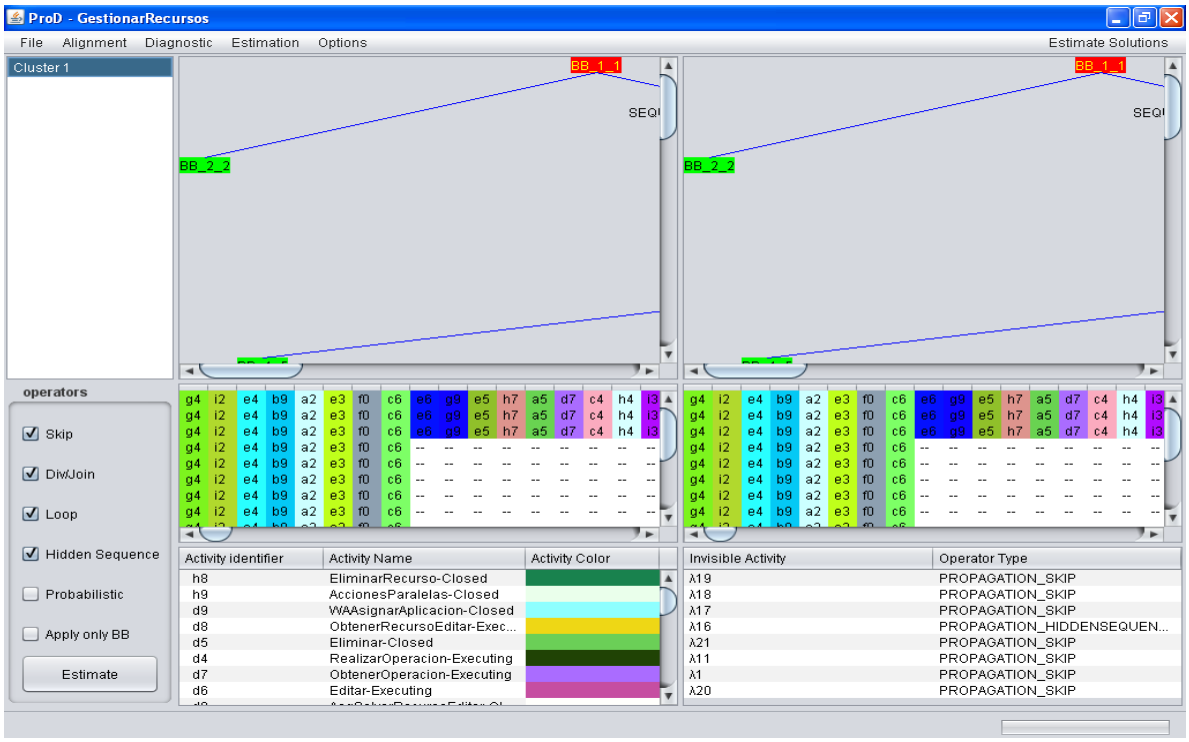

Figura 3.12. Detalle de la estimación de información ausente para el proceso Gestionar Recursos.

La Figura 3.12 muestra en el panel izquierdo, en la sección de configuración de los operadores (operators), los operadores seleccionados para la estimación (*Operador de salto*, *Operador de lazo*, *Operador de división/unión* y *Operador de secuencia oculta*). En el panel derecho, en la sección donde se describen las actividades invisibles, se muestran las actividades estimadas λ_1, λ_{11}, λ_{16}, λ_{17}, λ_{18}, λ_{19}, λ_{20} y λ_{21}. Siete de las ocho actividades estimadas se originaron por el *Operador de salto*, mientras que la restante se creó por el *Operador de secuencia oculta*.

Las actividades invisibles no aparecen numeradas de manera consecutiva porque en el proceso de estimación se originan 22 actividades invisibles, pero se eliminan del resultado final las actividades invisibles que solo tienen sentido localmente.

A partir de la estimación realizada se hace un análisis en conjunto con un especialista del negocio para garantizar que las actividades estimadas se manifiestan en el proceso ejecutado. De este análisis se concluye lo siguiente:

- Las actividades λ_{11} y λ_{21} indican casos incompletos (ausencia de actividades finales del proceso). Es decir, durante el pre-procesamiento de la alineación no se eliminaron del registro de evento los posibles casos incompletos, para así evaluar la respuesta del sistema desarrollado en estos casos. Cada una de las actividades estimadas se corresponde con la ausencia de información en casos diferentes.
- Las actividades λ_{1}, λ_{17}, λ_{18}, λ_{19}, y λ_{20} indican el éxito en los cinco subprocesos Agregar, Editar, Eliminar recursos, Condicionar Operación y Flujo de Negocio. En cada subproceso se almacenó un evento de fallo pero no aparece un evento que indique el éxito. Las actividades estimadas permitieron identificar fácilmente los casos en los que el proceso funcionó correctamente.
- La actividad λ_{16} permite delimitar el subproceso Condicionar Operación. Después del evento *CondicionarOperación-Execute* en uno de los casos no se registró ningún otro evento (caso incompleto), por lo cual, no es posible en el registro de evento analizado identificar adecuadamente el subproceso Condicionar Operación. El subproceso Condicionar Operación es el que incluye los subprocesos Agregar, Editar y Eliminar recursos. La actividad invisible en este caso permitió delimitar acertadamente el subproceso Condicionar Operación lo cual facilita el análisis de la información contenida en el registro de evento.

Como se puede apreciar cada una las actividades estimadas se corresponde con actividades existentes en el proceso de negocio ejecutado, aun cuando existen casos incompletos que pudiesen dificultar el análisis.

Las actividades estimadas facilitan la comprensión del proceso analizado al reflejar actividades ausentes e impedir que se establezcan incorrectas relaciones entre las actividades

existentes. Además, las actividades λ_1, λ_{17}, λ_{18}, λ_{19}, y λ_{20} resultan útiles para la posterior mejora del sistema implementado.

3.4 Otras aplicaciones del modelo EIAT

El modelo propuesto se aplicó también en el proceso Gestionar Roles informatizado con el Sistema Único de Identificación Nacional (SUIN), específicamente del módulo Gestionar Roles. En el análisis realizado de este proceso no solo se realizó la estimación de información ausente sino que, se empleó la descomposición del proceso con el objetivo de diagnosticar el proceso. El diagnóstico busca la detección de patrones interesantes, anomalías y desviaciones. En [143] se muestra el resultado de este análisis.

Además, la herramienta desarrollada a partir del modelo propuesto se utiliza en el análisis de los registros de evento del sistema de Planificación de Recursos Empresariales denominado Cedrux. En este caso se emplea con el objetivo de apoyar el desarrollo y soporte del sistema, dado que permite identificar posibles anomalías y el entendimiento de estas.

La herramienta desarrollada también se empleó en el análisis de los procesos Gestionar Cheques en el punto de venta Bar Vista al golfo y en el restaurant Aguiar, ambos locales pertenecientes al Hotel Nacional [153]. Para ambos registros de evento se estimó la información ausente y se determinaron las principales características de los procesos, lo cual apoyó la auditoría de dichos procesos.

3.5 Conclusiones del capítulo

En este capítulo se le dio cumplimiento a los objetivos tres y cuatro planteados para la investigación. Consecuentemente se desarrolló una aplicación informática considerando los aspectos definidos en el modelo EIAT y los algoritmos desarrollados para su aplicación. El propio desarrollo de la herramienta posibilitó detallar y mejorar los algoritmos propuestos, además constituyó la base para la validación del modelo desarrollado.

Para satisfacer el quinto objetivo se realizaron dos conjuntos de acciones, el primer conjunto dirigido a probar experimentalmente la propuesta realizada y el segundo conjunto se dirigió a la aplicación del modelo y la aplicación informática desarrollada en un entorno cubano.

Durante la validación experimental se seleccionaron cinco métricas que permiten medir las afectaciones que provoca la ausencia de información sobre la estructura y comprensión del modelo descubierto usando las técnicas desarrolladas en la minería de proceso.

Aun cuando existen registros de eventos diferentes se puede afirmar que la ausencia de información produce generalmente una reducción en las medidas de Fitness Unsatisfied, Fitness Unhandled, Precision y Non Fit Traces, que luego se recupera al estimar la información ausente utilizando el modelo propuesto. Esto demuestra la efectividad del modelo propuesto, el cual disminuye las afectaciones que produce la ausencia de información sobre la estructura y la comprensión de los modelos descubiertos usando minería de proceso.

La evaluación de la métrica ISA para todos los casos fue 1.0. Esto revela que los algoritmos aplicados, independientemente del momento, no introducen en el modelo descubierto problemas estructurales.

Las actividades estimadas a partir del registro de evento extraído del SUIN se validaron con la ayuda de un especialista del negocio y se comprobó la efectividad de la propuesta. La estimación realizada facilitó la comprensión del proceso analizado y sirvió como base para futuras mejoras del SUIN.

CONCLUSIONES

Se realizó un análisis del estado del arte sobre diferentes enfoques en el tratamiento de la ausencia de información en las trazas usadas en la minería de proceso. A partir de este análisis se propusieron objetivos que fueron cumplidos y se arriba a las siguientes conclusiones:

- Se describen dos nuevas manifestaciones de ausencia de información (*Secuencia oculta de subprocesos* y *Opciones equiprobables*) que no se habían reportado en la literatura. Sus descripciones permiten hacer un análisis más completo del tratamiento de la ausencia de información.
- El modelo EIAT posibilita inicialmente la descomposición del proceso analizado. La descomposición se realiza considerando los patrones básicos de flujo de trabajo. Estos patrones se identificaron a nivel de subprocesos lo cual contribuye a la posterior identificación de las manifestaciones de ausencia de información.
- El modelo EIAT, a partir de la descomposición del proceso obtenida y de un conjunto de operadores definidos, posibilita la identificación de las situaciones de ausencia de información y la estimación de la información ausente.
- Las pruebas realizadas haciendo uso de la aplicación informática desarrollada arrojaron que, aun cuando existen registros de eventos diferentes, la ausencia de información produce generalmente una reducción en las medidas de Fitness Unsatisfied, Fitness Unhandled, Precision y Non Fit Traces, que luego se recupera al estimar la información ausente utilizando el modelo EIAT. El modelo propuesto no introduce afectaciones estructurales, lo cual se evidencia en los resultados obtenidos para la métrica ISA.

- Con los resultados obtenidos en la evaluación, con la aplicación del modelo EIAT y la herramienta desarrollada en diferentes entornos, queda demostrada la efectividad de la propuesta, la cual disminuye las afectaciones que produce la ausencia de información sobre la estructura y la comprensión de los modelos descubiertos usando minería de proceso.

RECOMENDACIONES

Al concluir la presente investigación aun quedan un conjunto de acciones que se proponen a manera de recomendación:

- Desde el punto de vista teórico seguir profundizando en la obtención de un mecanismo de descomposición del proceso que sea más flexible. En este caso se puede valorar problemas relacionados con la completitud del registro de evento.
- Desde el punto de vista teórico seguir profundizando en la obtención de un mecanismo de alineación de las trazas más ajustados a la descomposición de procesos.
- Profundizar desde el punto de vista teórico en la identificación del significado de las actividades estimadas en dependencia del contexto. Haciendo uso de los registros de evento con descripción semántica y el análisis de casos semejantes especialmente en casos en lo que se evidencie ruido.

REFERENCIAS BIBLIOGRÁFICAS

[1] IBM, "A New Way of Working: Insights from Global Leaders," IBM Institute for Business Value, United States of America, 2010.

[2] W. M. P. v. d. Aalst and K. M. v. Hee, *Workflow Management: Models, Methods, and Systems*. USA, 2004. ISBN: 0-262-01189-1.

[3] J. B. Hill*, et al.*, "Magic Quadrant for Business Process Management Suites," Gartner, Inc. February 2009.

[4] DAA, "INFORME ESPECIAL DE IT-LATINO.NET," DAA CONTENIDOS DIGITALES, S.L., Barcelona, España, 2010.

[5] M. Weske, *Business Process Management. Concepts, Languages, Architectures*, 2007. ISBN: 978-3-540-73521-2.

[6] K. B. Hendricks*, et al.*, "The impact of enterprise systems on corporate performance: A study of ERP, SCM, and CRM system implementations," *Operations Management,* vol. 25, pp. 65-82, January, 2007. ISSN: 0272-6963. DOI: 10.1016/j.jom.2006.02.002.

[7] C. D. Tarantilis*, et al.*, "A Web-based ERP system for business services and supply chain management: Application to real-world process scheduling," *European Journal of Operational Research,* vol. 187 pp. 1310-1326, June, 2008.

[8] M.-K. Chang*, et al.*, "Understanding ERP system adoption from the user's perspective," *International Journal of Production Economics,* vol. 113, pp. 928-942, June, 2008.

[9] SAP. (2010, April 2011). Available: http://www.sap.com

[10] J. E. Cook and A. L. Wolf., "Automating Process Discovery Through Event-Data Analysis," in *ICSE '95: Proceedings of the 17th international conference on Software engineering*, New York, USA, 1995, pp. 1-10. ISBN: 0-89791-708-1. DOI: 10.1145/225014.225021.

[11] J. E. Cook, "Process Discovery and Validation Through Event-Data Analysis," PhD thesis, 1996.

[12] R. Agrawal*, et al.*, "Mining Process Models from Workflow Logs," in *EDBT '98 Proceedings of the 6th International Conference on Extending Database Technology: Advances in Database Technology*, 1998, pp. 1-15. ISBN: 3-540-64264-1.

[13] W. M. P. v. d. Aalst, *Process Mining. Discovery, Conformance and Enhancement of Business Processes*: Springer Heidelberg Dordrecht London New York, 2011. ISBN: 978-3-642-19344-6. DOI: 10.1007/978-3-642-19345-3.

[14] W. M. P. V. d. Aalst*, et al.*, "Process Mining Manifesto," *Business Process Management Workshops 2011, Lecture Notes in Business Information Processing.Springer-Verlag,* vol. 99, 2011.

[15] R. L. Sallam*, et al.*, "Magic Quadrant for Business Intelligence Platforms," Gartner, Inc. 2011.

[16] W. M. P. v. d. Aalst and A. J. M. M. Weijters, "Process Mining: A Research Agenda," *Special Issue of Computers in Industry, Elsevier Science Publishers, Amsterdam,* vol. 53, 2004. DOI: 10.1016/j.compind.2003.10.001.

[17] W. M. P. v. d. Aalst*, et al.*, "ProcessMining: A Two-Step Approach to Balance Between Underfitting and Overfitting," *Software and Systems Modeling,* vol. 9, pp. 87-111, 2009. DOI: 10.1007/s10270-008-0106-z.

[18] W. M. P. v. d. Aalst*, et al.*, "Workflow Mining: Discovering process models from event logs," *IEEE Transactions on Knowledge and Data Engineering,* vol. 16, pp. 1128-1142, 2004. DOI: 10.1109/TKDE.2004.47.

[19] W. M. P. V. d. Aalst*, et al.*, "Process mining and verification of properties: An approach based on temporal logic," in *Proc. OTM Conferences 05*. LNCS, 2005, pp. 130 -147.

[20] A. Adriansyah*, et al.*, "Towards Robust Conformance Checking," in *BPM 2010 Workshops, Proceedings of the 6th Workshop on Business Process Intelligence (BPI2010)*, Lecture Notes in Business Information Processing. Springer, Berlin, vol. 66, pp. 122-133, 2011. ISBN: 978-3-642-20510-1. DOI: 10.1007/978-3-642-20511-8_11.

[21] A. Rozinat and W. M. P. v. d. Aalst, "Conformance checking of processes based on monitoring real behavior," *Inf. Syst.,* vol. 33, pp. 64-95, 2008. ISSN: 0306-4379. DOI: 10.1016/j.is.2007.07.001.

[22] W. M. P. v. d. Aalst*, et al.*, "Auditing 2.0: Using Process Mining to Support Tomorrow's Auditor," *IEEE Computer Society, vol.* 43*, pp.* 90 - 93, 2010. ISSN: 0018-9162. DOI: 10.1109/MC.2010.61.

[23] A. K. A. d. Medeiros, "Genetic Process Mining," PhD. thesis, Technische Universiteit Eindhoven, 2006.

[24] B. v. Dongen, "Process Mining and Verification," PhD Thesis, Technische Universiteit Eindhoven, 2007.

[25] W. M. P. V. D. Aalst*, et al.*, "Workflow Patterns," *Distrib. Parallel Databases,* vol. 14, pp. 5-51, 2003. ISSN: 0926-8782. DOI: 10.1023/a:1022883727209.

[26] W. M. P. v. d. Aalst*, et al.*, "Business Process Management: A Survey " *Lecture Notes in Computer Science,* vol. 2678, pp. 1-12, 2003.

[27] J. D. Weerdt*, et al.*, "A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs," *Inf. Syst.,* vol. 37, pp. 654-676, 2012. ISSN: 0306-4379. DOI: 10.1016/j.is.2012.02.004.

[28] W. M. P. v. d. Aalst*, et al.*, "Business process mining: An industrial application," *Information Systems,* vol. 32, pp. 713-732, 2007. ISSN: 0306-4379. DOI: 10.1016/j.is.2006.05.003.

[29] R. P. J. C. Bose and W. M. P. v. d. Aalst, "Context Aware Trace Clustering: Towards Improving Process Mining Results," in *International Conference on Data Mining*, 2009, pp. 401-412. ISBN: 978-3-642-12186-9. DOI: 10.1007/978-3-642-12186-9_16.

[30] M. Song*, et al.*, "Trace Clustering in Process Mining," in Business Process Management Workshops, 2010, vol. 17, pp. 109-120. ISBN: 978-3-642-00327-1.

[31] W. M. P. v. d. Aalst and C. W. Günther, "Finding Structure in Unstructured Processes: The Case for Process Mining," in *ACSD '07 Proceedings of the Seventh International Conference on Application of Concurrency to System Design*, 2007, pp. 7-10. ISBN: 0-7695-2902-X.

[32] L. T. Ly*, et al.*, "Data Transformation and Semantic Log Purging for Process Mining. ," in *24th International Conference on Advanced Information Systems Engineering (CAiSE'12)*, Gdansk, Poland, 2012.

[33] R. Z. Farkhady and S. H. Aali, "A Probabilistic Approach for Process Mining in Incomplete and Noisy Logs," in *International MultiConference of Engineers and Computer Scientists (IMECS 2011)*, vol. 2188, pp. 415-420, 2011. ISSN: 20780966.

[34] F. Folino*, et al.*, "Discovering expressive process models from noised log data," presented at the Proceedings of the 2009 International Database Engineering & Applications Symposium, Cetraro - Calabria, Italy, ACM, pp. 162-172, 2009. ISBN: 978-1-60558-402-7. DOI: 10.1145/1620432.1620449.

[35] A. Tiwari*, et al.*, "A review of business process mining: state-of-the-art and future trends," *Business Process Management,* vol. 14, pp. 5-22, 2008. ISSN: 1463-7154. DOI: 10.1108/14637150810849373.

[36] A. Adriansyah*, et al.*, "Cost-Based Conformance Checking using the A* Algorithm," BPM Center Report BPM-11-11, BPMcenter.org, 2011.

[37] J. Muñoz-Gama and J. Carmona, "A fresh look at Precision in Process Conformance," in *8th international conference on Business process management*, Hoboken, NJ, USA, 2010, pp. 211-226. ISBN: 3-642-15617-7.

[38] L. Wen*, et al.*, "A Novel Approach for Process Mining Based on Event Types," *BETA Working Paper Series, WP 118, Eindhoven University of Technology, Eindhoven,* 2004.

[39] R. Bergenthum*, et al.*, "Process Mining Based on Regions of Languages," *Lecture Notes in Computer Science*, vol. 4714, vol. 4714, pp. 375-383, 2007. DOI: 10.1007/978-3-540-75183-0_27.

[40] J. M. Werf*, et al.*, "Process Discovery Using Integer Linear Programming," presented at the Proceedings of the 29th international conference on Applications and Theory of Petri Nets, Xi'an, China, Springer-Verlag, pp. 368-387, 2008. DOI: 10.1007/978-3-540-68746-7_24.

[41] R. P. J. C. Bose and W. M. P. v. d. Aalst, "Trace Alignment in Process Mining: Opportunities for Process Diagnostics," in *International Conference on*

Business Process Management (BPM'2010), 2010, pp. 227-242. ISBN: 3-642-15617-7.

[42] A. Rozinat, et al., "The Need for a Process Mining Evaluation Framework in Research and Practice. ," in *BPM 2007 International Workshops (BPI, BPD, CBP, ProHealth, RefMod, Semantics4ws)*. Lecture Notes in Computer Science. Springer, Berlin, 2008, vol. 4928, pp. 84-89. ISBN: 3-540-78237-0 978-3-540-78237-7.

[43] R. P. J. C. Bose and W. M. P. v. d. Aalst, "Abstractions in Process Mining: A Taxonomy of Patterns." Lecture Notes in Computer Science, vol. 5701, U. Dayal*, et al.*, Eds., ed: Springer Berlin / Heidelberg, 2009, pp. 159-175. ISBN: 978-3-642-03847-1. DOI: 10.1007/978-3-642-03848-8_12.

[44] C. W. G¨unther*, et al.*, "Activity Mining by Global Trace Segmentation," in *Business Process Management Workshops*. Lecture Notes in Business Information Processing, vol. 43, pp. 128-139, 2009. ISBN: 978-3-642-12185-2. DOI: 10.1007/978-3-642-12186-9_13.

[45] G. Schimm, "Mining Most Specic Workflow Models from Event-Based Data," in *International Conference on Business Process Management (BPM 2003).* Lecture Notes in Computer Science, vol. 2678, pp. 25-40, 2003. ISBN: 3-540-40318-3.

[46] G. Schimm, "Mining Exact Models of Concurrent Workflows," *Computers in Industry,* vol. 53, pp. 265-281, 2004. DOI: 10.1016/j.compind.2003.10.003.

[47] A. J. M. M. Weijters and J. T. S. Ribeiro, "Flexible Heuristics Miner (FHM) ," *BETA Working Paper Series, WP 334, Eindhoven University of Technology, Eindhoven,* 2010.

[48] C. W. Günther and W. M. P. v. d. Aalst, "Fuzzy Mining: Adaptive Process Simplification Based on Multi-Perspective Metrics," in *International Conference on Business Process Management (BPM 2007)*. Lecture Notes in Computer Science, vol. 4714, pp. 328-343, 2007. ISBN: 3-540-75182-3, 978-3-540-75182-3.

[49] R. Yzquierdo*, et al.*, "Operadores para el tratamiento de ausencia de información en la minería de procesos," in *UCiencia*, Habana, Cuba, 2012.

[50] R. Yzquierdo*, et al.*, "Modelo para la estimación de información ausente en la minería de proceso," presented at the XVI FORUM DE CIENCIA Y TÉCNICA, Habana, Cuba, 2012.

[51] S. Workflow Management Coalition, *Workflow Management Coalition, Terminology & Glossary (Document No. WFMC-TC-1011)*: Workflow Management Coalition Specification, 1999. DOI: citeulike-article-id:6080985.

[52] H. Smith and P. Fingar, *Business Process Management (BPM):The Third Wave.* Meghan-Kiffer Press, 2003. ISBN: 0929652339.

[53] S. Chaudhuri*, et al.*, "An overview of business intelligence technology," *Commun. ACM,* vol. 54, pp. 88-98, 2011. ISSN: 0001-0782. DOI: 10.1145/1978542.1978562.

[54] S. Rizzi, "Collaborative business intelligence " in *Business Intelligence - First European Summer School (eBISS 2011)*. Tutorial Lectures, M.-A. Aufaure and E. Zimanyi (Eds.), LNBIP 96, Springer, pp. 186-205, 2012.

[55] M. Safeer and S. Zafar, "Impact of business intelligence competency center in success/ failure of B.I. applications," in *Multitopic Conference (INMIC), 2011 IEEE 14th International* Karachi, Pakistan, pp. 267 – 272, 2011. ISBN: 978-1-4577-0654-7. DOI: 10.1109/INMIC.2011.6151486.

[56] D. K. Folinas*, et al.*, "In-field logistics processes management based on business activities monitoring systems paradigm " *International Journal of Logistics Systems and Management,* vol. 8, pp. 1-18, 2010. ISBN: 1742-7975. DOI: 10.1504/IJLSM.2011.037416.

[57] D. T. Goomas*, et al.*, "Business Activity Monitoring: Real-Time Group Goals and Feedback Using an Overhead Scoreboard in a Distribution Center," *Journal of Organizational Behavior Management* vol. 31, pp. 196-209, 2011. DOI: 0.1080/01608061.2011.589715.

[58] D. Wang*, et al.*, "Active Complex Event Processing infrastructure: Monitoring and reacting to event streams " in *Data Engineering Workshops (ICDEW), 2011 IEEE 27th International Conference*, Hannover, USA, pp. 249 - 254, 2011. ISBN: 978-1-4244-9194-0. DOI: 10.1109/ICDEW.2011.5767635.

[59] M. Roth and S. Donath, "Applying Complex Event Processing towards Monitoring of Multi-party Contracts and Services for Logistics – A Discussion," in *Business Process Management Workshops*. Lecture Notes in Business Information Processing, vol. 99, pp. 458-463, 2012. DOI: 10.1007/978-3-642-28108-2_44.

[60] A.-W. Sheer*, et al.*, *Corporate performance management: ARIS in practice*. Springer, pp. 6-39, 2006. ISBN: 10 3-540-30703-6.

[61] B. Paladino, *Innovative corporate performance management: Five key principles to accelerate results*. John Wiley & Sons, pp. 35-47, 2010. ISBN: 978-0-470-62773-0.

[62] L. Wang*, et al.*, "Continuous Process Improvement in Banking Sector and a Model Design for Performance Enhancement," *International Journal of Business and Management,* vol. 7, 2012. DOI: 10.5539/ijbm.v7n2p130.

[63] L. Chen*, et al.*, "Business Process Continuous Improvement System Based on Workflow Mining Technology," in *Computer Science and Information Engineering, 2009 WRI World Congress*, Los Angeles, CA, pp. 414 - 418, 2009. ISBN: 978-0-7695-3507-4. DOI: 10.1109/CSIE.2009.487.

[64] F. Abou Moghdeb*, et al.*, "Business process improvement and organizational theory: The missing link," in *Information Resources Management Assocation (IRMA) International Conference*, Vancouver, British Columbia. IGI Publishing, pp. 253-264, 2007. ISBN: 978-1-59904-929-8.

[65] G. Zellner, "A structured evaluation of business process improvement approaches," *Business Process Management Journal,* vol. 17, pp. 203 - 237, 2011. ISSN: 1463-7154. DOI: 10.1108/14637151111122329.

[66] E. Soltani*, et al.*, "A review of the theory and practice of managing TQM: An integrative framework," *Total Quality Management & Business Excellence* vol. 19, pp. 461-479, 2008. DOI: 10.1080/14783360802018103.

[67] A. M. H. Gorji and J. A. Farooquie, "A comparative study of total quality management of health care system in India and Iran," *BMC Research Notes,* vol. 4, 2011. DOI: 10.1186/1756-0500-4-566.

[68] R. G. Schroeder*, et al.*, "Six Sigma: Definition and underlying theory," *Journal of Operations Management,* vol. 26, pp. 536-554, 2008. ISSN: 0272-6963. DOI: 10.1016/j.jom.2007.06.007.

[69] D. Grigori*, et al.*, "Business Process Intelligence," *Computers in Industry,* vol. 53, pp. 321-343, 2004.

[70] Software-AG. (2011, September 2011). *ARIS Process Performance Manager*. Available:

http://www.softwareag.com/corporate/products/aris_platform/aris_controlling/aris_process_performance/overview/default.asp

[71] PYMNTS. (2011, September 2011). *OpenConnect Comprehend Business Process Intelligence and Analytics Software Increases Productivity, Promotes Job Creation*. Available: http://pymnts.com/news/businesswire-feed/2011/march/21/openconnect-comprehend-business-process-intelligence-and-analytics-software-increases-productivity-promotes-job-creation-20110321005124/

[72] Stereologic-Ltd. (2012, January 2012). *Discover, Visualize and Analyze Business Processes in Real Time*. Available: http://www.stereologic.com/

[73] Fourspark. (2012, January 2012). *Products*. Available: http://fourspark.no/?page_id=11

[74] Futura-Process-Intelligence. (2012, January 2012). *Futura Process Intelligence*. Available: http://biz.prlog.org/Futura/

[75] FUJITSU. (2012, January 2012). *Discover & Visualize Business Processes – Automatically*. Available: http://www.fujitsu.com/global/services/software/interstage/solutions/bpmgt/bpm-services/apd/

[76] Exeura. (2012, January 2012). *Exeura Homepage*. Available: http://www.exeura.com/home.php?lan=en

[77] Lontas. (2012, January 2012). *Process Discovery Focus*. Available: http://www.iontas.com/pages/products/pdf.php

[78] QPR. (2012, January 2012). *Automated Business Process Discovery Software QPR ProcessAnalyzer*. Available: http://www.qpr.com/products/qpr-processanalyzer.htm

[79] W. M. P. v. d. Aalst*, et al.*, "ProM: The Process Mining Toolkit," in *Proceedings of BPM (Demos)'2009*, Ulm, Germany. CEUR-WS.org, vol. 489, 2009.

[80] UPC. (2010, January 2011). *DBMINER*. Available: http://personals.ac.upc.edu/msole/homepage/dbminer.html

[81] Perceptive-Software. (2012, January 2012). *Perceptive Reflect*. Available: http://www.perceptivesoftware.com/products/product-explorer/business-process/perceptive-reflect.psi

[82] W. M. P. v. d. Aalst*, et al.*, "Process Mining: A Two-Step Approach using Transition Systems and Regions," BPM Center Report BPM-06-30, BPMcenter.org, 2006.

[83] L. Wen*, et al.*, "A novel approach for process mining based on event types," *J. Intell. Inf. Syst.,* vol. 32, pp. 163-190, 2009. ISSN: 0925-9902. DOI: 10.1007/s10844-007-0052-1.

[84] R. P. J. C. Bose and W. M. P. v. d. Aalst, "Process diagnostics using trace alignment: Opportunities, issues, and challenges," *Inf. Syst.,* vol. 37, pp. 117-141, 2012. ISSN: 0306-4379. DOI: 10.1016/j.is.2011.08.003.

[85] C. W. Günther*, et al.*, "Using Process Mining to Learn from Process Changes in Evolutionary Systems," *International Journal of Business Process Integration and Management*, vol. 3, pp. 61-78, 2008. DOI: 10.1504/IJBPIM.2008.019348.

[86] C. W. Günther. (2009, February 2011). *XES Standard Definition*. Available: www.xes-standard.org

[87] C. W. Günther, "Process Mining in Flexible Environments," Ph.D. thesis, Eindhoven University of Technology, Eindhoven, 2009.

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

[89] S. Goedertier*, et al.*, "Process Mining as First-Order Classification Learning on Logs with Negative Events," in *BPM 2007 International Workshops (BPI, BPD, CBP, ProHealth, RefMod, Semantics4ws).* Lecture Notes in Computer Science, vol. 4928, pp. 42–53, 2008.

[90] S. Goedertier*, et al.*, "Robust Process Discovery with Artificial Negative Events," *Journal of Machine Learning Research*, vol. 10, pp. 1305-1340, 2009. ISSN: 1532-4435. DOI: 10:1305–1340.

[91] A. Rozinat*, et al.*, "The Need for a Process Mining Evaluation Framework in Research and Practice. ," in *BPM 2007 International Workshops (BPI, BPD, CBP, ProHealth, RefMod, Semantics4ws)*. Lecture Notes in Computer Science, vol. 4928, pp. 84-89, 2008. ISBN: 3-540-78237-0 978-3-540-78237-7.

[92] A. J. M. M. Weijters and W. M. P. v. d. Aalst, "Rediscovering Workflow Models from Event-Based Data using Little Thumb," *Integrated Computer-Aided Engineering,* vol. 10, pp. 151-162, 2003. ISSN: 1069-2509.

[93] W. M. P. V. d. Aalst, "Three Good Reasons for Using a Petri-net-based Workflow Management System," in *The International Working Conference on Information and Process Integration in Enterprises (IPIC'96)*, pp. 179-201, 1996.

[94] W. M. P. v. d. Aalst, "Structural Characterizations of Sound Workflow Nets," *Computing Science Reports 96/23,* pp. 179-201, 1996.

[95] B. F. v. Dongen and W. M. P. v. d. Aalst, "Multi-phase Process Mining: Building Instance Graphs," *Lecture Notes in Computer Science,* vol. 3288, pp. 362-376, 2004.

[96] V. Rubin, "A Workflow Mining Approach for Deriving Software Process Models," PhD thesis, University of Paderborn, 2007.

[97] A. Rozinat and W. M. P. v. d. Aalst, "Conformance checking of processes based on monitoring real behavior," *Inf. Syst.*, vol. 33, pp. 64-95, 2008. ISSN: 0306-4379. 10.1016/j.is.2007.07.001.

[98] J. Mendling*, et al.*, "Understanding the Occurrence of Errors in Process Models Based on Metrics," in *Proceedings of the OTM Conference on Cooperative Information Systems (CoopIS 2007)*. Lecture Notes in Computer Science, vol. 4803, pp. 113-130, 2007.

[99] A. Adriansyah*, et al.*, "Conformance Checking Using Cost-Based Fitness Analysis," in *Enterprise Distributed Object Computing Conference (EDOC), 2011 15th IEEE International* Helsinki, pp. 55 – 64, 2011. ISBN: 978-0-7695-4425-0. 10.1109/EDOC.2011.12.

[100] W. M. P. v. d. Aalst, "Business Process Simulation Revisited," in *Enterprise and Organizational Modeling and Simulation*. Lecture Notes in Business Information Processing, vol. 63, pp. 1–14, 2010.

[101] W. M. P. v. d. Aalst*, et al.*, "Time Prediction Based on Process Mining," *Information Systems,* vol. 36, pp. 450–475, 2011.

[102] W. M. P. v. d. Aalst*, et al.*, "Soundness of Workflow Nets: Classification, Decidability and Analysis," *Formal Aspects of Computing,* vol. 23, pp. 333-363, 2011. ISSN: 0934-5043. DOI: 10.1007/s00165-010-0161-4.

[103] (2012, February 2012). *Workflow Patterns Home Page*. Available: http://www.workflowpatterns.com

[104] W. M. P. v. d. Aalst and A. H. M. t. Hofstede, "Workflow patterns put into context," *Software and Systems Modeling,* pp. 1-5, 2012. DOI: 10.1007/s10270-012-0233-4.

[105] B. F. v. Dongen*, et al.*, "Process Mining: Overview and Outlook of Petri Net Discovery Algorithms " in *Transactions on Petri Nets and Other Models of Concurrency II*. Lecture Notes in Computer Science, vol. 5460, pp. 225-242, 2009. DOI: 10.1007/978-3-642-00899-3_13.

[106] W. M. P. v. d. Aalst, "On the Representational Bias in Process Mining," presented at the Proceedings of the 2011 IEEE 20th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises. IEEE Computer Society, pp. 2-7, 2011. DOI: 10.1109/wetice.2011.64.

[107] Eindhoven-University-of-Technology, "ProM," ed, 2012. Available: http://www.processmining.org/prom/start

[108] J. E. Cook*, et al.*, "Discovering Models of Behavior for Concurrent Workflows," *Computers in Industry,* pp. 297-319, 2004.

[109] L. Wen*, et al.*, "Detecting Implicit Dependencies Between Tasks from Event Logs," *APWeb, Lecture Notes in Computer Science. Springer,* vol. 3841, pp. 591-603, 2006.

[110] J. E. Cook and A. L. Wolf, "Discovering Models of Software Processes from Event-Based Data," *ACM Transactions on Software Engineering and Methodology,* pp. 215-249, 1998. ISSN: 1049-331X. DOI: 10.1145/287000.287001.

[111] J. E. Cook and A. L. Wolf, "Event-Based Detection of Concurrency," in *Proceedings of the Sixth International Symposium on the Foundations of Software Engineering (FSE-6)*, New York, NY, USA, pp. 35-45, 1998.

[112] J. Herbst, "A Machine Learning Approach to Workflow Management," in *11th European Conference on Machine Learning*. Springer-Verlag, pp. 183 - 194, 2000. ISBN: 3-540-67602-3.

[113] J. Herbst, "Ein induktiver Ansatz zur Akquisition und Adaption von Workflow-Modellen," PhD thesis, University Ulm, 2001.

[114] J. Herbst and D. Karagiannis, "Integrating Machine Learning and Work-flow Management to Support Acquisition and Adaptation of Workflow Models," *International Journal of Intelligent Systems in Accounting, Finance and Management,* pp. 67-92, 2000.

[115] J. Herbst and D. Karagiannis, "Workflow Mining with InWoLvE.," *Computers in Industry,* vol. 53, pp. 245-264, 2004.

[116] G. Schimm, "Process Miner - A Tool for Mining Process Schemes from Event-based Data," in *8th European Conference on Articial Intelligence (JELIA)*, Berlin. Lecture Notes in Computer Science, vol. 2424, pp. 525-528, 2002.

[117] G. Schimm, "Generic Linear Business Process Modeling," in *the ER 2000 Workshop on Conceptual Approaches for E-Business and The World Wide Web and Conceptual Modeling*, Lecture Notes in Computer Science. Springer-Verlag, Berlin. Lecture Notes in Computer Science, vol. 1921, pp. 31-39, 2000.

[118] G. Schimm (2004, January 2011). *Process Mining*. Available: http://www.processmining.de/

[119] G. Greco*, et al.*, "Mining Hierarchies of Models: From Abstract Views to Concrete Specications," *Business Process Management,* vol. 3649, pp. 32-47, 2005.

[120] G. Greco*, et al.*, "Mining Expressive Process Models by Clustering Workflow Traces," *PAKDD. Lecture Notes in Computer Science. Springer,* vol. 3056, pp. 52-62, 2004.

[121] W. M. P. v. d. Aalst and A. J. M. M. Weijters, "Chapter 10: Process Mining," *Process-Aware Information Systems: Bridging People and Software Through Process Technology. John Wiley & Sons Inc,* 2005.

[122] W. M. P. v. d. Aalst and A. J. M. M. Weijters, "Process Mining," *Special Issue of Computers in Industry. Elsevier Science Publishers, Amsterdam,* vol. 53, 2004.

[123] B. F. v. Dongen and W. M. P. v. d. Aalst, "Multi-phase Process mining: Aggregating Instance Graphs into EPCs and Petri Nets," in *Second International Workshop on Applications of Petri Nets to Coordination, Workflow and Business Process Management(PNCWB)*, 2005.

[124] B. F. v. Dongen*, et al.*, "Verication of EPCs: Using Reduction Rules and Petri Nets," *CAiSE, Lecture Notes in Computer Science.Springer,* vol. 3520, pp. 372-386, 2005.

[125] R. Lorenz*, et al.*, "How to synthesize nets from languages: a survey," presented at the Proceedings of the 39th conference on Winter simulation: 40 years! The best is yet to come, Washington D.C. IEEE Press, pp. 637-647, 2007.

[126] T. v. d. Wiel, "Process Mining using Integer Linear Programming," Masters, Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, 2010.

[127] M. Song and W. M. P. V. d. Aalst, "Supporting process mining by showing events at a glance," in *17th Annual Workshop on Information Technologies and Systems (WITS)*, pp. 139 -145, 2007.

[128] R. P. J. M. v. Arendonk, "A Benchmark Set for Process Discovery Algorithms," Master, Mathematics & Computer Science, Eindhoven University of Technology, Eindhoven, 2011.

[129] A. Rozinat*, et al.*, "Towards an Evaluation Framework for Process Mining Algorithms," *BPM Center Report BPM-07-06, BPMcenter.org,* 2007.

[130] J. D. Weerdt*, et al.*, "A critical evaluation study of model-log metrics in process discovery," in *6th International Workshop on Business Process Intelligence*. Springer, vol. 66, pp. 158-169, 2010. ISBN: 978-3-642-20510-1.

[131] J. Muñoz-Gama and J. Carmona, "A fresh look at precision in process conformance," in *8th international conference on Business process management*, Hoboken, NJ, USA. Springer-Verlag, pp. 211-226, 2010. ISBN: 3-642-15617-7.

[132] A. Rozinat and W. M. P. v. d. Aalst, "Conformance Testing: Measuring the Fit and Appropriateness of Event Logs and Process Models," presented at the Third International Conference on Business Process Management(BPM 2005), France, 2005.

[133] R. P. J. C. Bose and W. M. P. V. d. Aalst, "Trace Clustering Based on Conserved Patterns: Towards Achieving Better Process Models " in *Proceedings of the 5th International Workshop on Business Process*

Intelligence. Lecture Notes in Business Information Processing, vol. 43, pp. 170-181, 2010. ISBN: 978-3-642-12186-9. DOI: 10.1007/978-3-642-12186-9_16.

[134] G. Greco*, et al.*, "Mining Hierarchies of Models: From Abstract Views to Concrete Specifications," in *Business Process Management*. Lecture Notes in Computer Science, vol. 3649, pp. 32-47, 2005. DOI: 10.1007/11538394_3.

[135] G. Greco*, et al.*, "Mining taxonomies of process models," *Data & Knowledge Engineering,* vol. 67, pp. 74-102, 2008.

[136] J. Li*, et al.*, "Mining Context-Dependent and Interactive Business Process Maps using Execution Patterns," in *BPM 2010 Workshops. Springer-Verlag*, vol. 66, pp. 109 -121, 2010.

[137] O. Gotoh, "Multiple Sequence Alignment: Algorithms and Applications," *Advanced Biophysics,* vol. 36, pp. 159-206, 1999.

[138] D. Feng and R. Doolittle, "Progressive Sequence Alignment as a Prerequisite to Correct Phylogenetic Trees," *Journal of Molecular Evolution,* vol. 25, pp. 351-360, 1987.

[139] R. D. D. Feng, "Progressive Alignment of Amino Acid Sequences and Construction of Phylogenetic Trees from Them," *Methods in Enzymology,* vol. 266, pp. 368-382, 1996.

[140] C. Daniel*, et al.*, "PFAAT version 2.0: A Tool for Editing, Annotating, and Analyzing Multiple Sequence Alignments," *BMC Bioinformatics,* vol. 8, 2007. DOI: 10.1186/1471-2105-8-381.

[141] C. Notredame, "Recent Progress in Multiple Sequence Alignment: A Survey," *Pharmacogenomics,* vol. 3, pp. 131-144, 2002.

[142] G. Greco*, et al.*, "Discovering Expressive Process Models by Clustering Log Traces," *IEEE Transaction on Knowledge and Data Engineering,* vol. 18, pp. 1010–1027, 2006.

[143] R. Yzquierdo*, et al.*, "Diagnóstico de proceso basado en el descubrimiento de subprocesos," *Revista Ingeniería Industrial,* vol. 33, pp. 133-141, 2012.

[144] R. Yzquierdo*, et al.*, "Propuesta para el diagnóstico de los procesos en la gestión empresarial," presented at the *Taller Científico XXIV Aniversario del CETED*, Habana, Cuba, 2012.

[145] R. Yzquierdo*, et al.*, "Sistema para el diagnóstico del proceso basado en su descomposición," presented at the *VII Peña Tecnológica del MININT*, Habana, Cuba, 2012.

[146] R. Yzquierdo*, et al.*, "Propuesta para el diagnóstico de los procesos empresariales utilizando minería de procesos," presented at the *XII Conferencia Internacional de Ciencias Económicas y Empresariales*, Camagüey, Cuba, 2012.

[147] R. Yzquierdo*, et al.*, "Diagnóstico del proceso: una herramienta para la auditoría," presented at the VIII Encuentro Internacional de Contabilidad, Auditoría y Finanzas, Habana, Cuba, 2012.

[148] R. Yzquierdo, *et al.*, "Minería de procesos como herramienta para la auditoría," Ciencias de la Información, 2012 (Aceptado).

[149] R. Yzquierdo, et al., "Sub-process discovery: Opportunities for Process Diagnostics," in CONFENIS, Bélgica, 2012.

[150] R. Seidel and M. Sharir, "Top-Down Analysis of Path Compression," *SIAM J. Comput.,* vol. 34, pp. 515-525, 2005.

[151] Process-Mining-group, "Process Log Generator," in *Process Mining group*, ed. Padua, Italy, 2011.

[152] Microsoft. (2012, January 2012). Windows Workflow Foundation. Available: http://msdn.microsoft.com/es-es/netframework/aa663328

[153] L. González and M. Suárez, "Procedimiento para la aplicación de técnicas de minería de proceso en las auditorías de procesos," Grado, Facultad de Ingeniería Industrial, Instituto Superior Politécnico José Antonio Echeverría, Habana, Cuba, 2012.

Anexos

Anexo 1 Algoritmo para la corrección de las actividades repetidas

Algoritmo Actividades-Repetidas

Entrada: Matriz alineada *M*

Salida: Matriz alineada

1: **Para** cada columna de *M* **Hacer**

2: **Si** la columna tiene más de una actividad **Entonces**

3: Se almacena el índice de la columna en una lista *L*

FinSi

FinPara

4: **Para** cada elemento *c* de *L* **Hacer** //se recorre de mayor a menor

5: **Para** cada actividad de la columna *c* **Hacer**

6: Se crea una lista temporal *T*(la cantidad de elementos es igual a la cantidad de filas de *M*) que contiene la actividad seleccionada en las mismas posiciones que la columna original y el resto de las posiciones se completan con "-"

7: Se agrega a *M* una nueva columna a partir de la lista *T*, en la posición siguiente a la de la columna original

FinPara

8: Se elimina de *M* la columna analizada

FinPara

9: Devolver *M*

Anexo 2 Algoritmo para la búsqueda de secuencias

Algoritmo Buscar secuencia

Entrada: Bloque de construcción $C^i_{\mathcal{A}}$

Salida: Lista de bloques de construcción

1: Crear dos listas vacías *ListaSeparador* y *ListaHijos*. *ListaSeparador* tiene como objetivo almacenar el identificador de las columnas que delimitan cada bloque de construcción. *ListaHijos* almacena los bloques de construcción obtenidos en una posible descomposición

2: **Para** cada columna de $C^i_{\mathcal{A}}$ **Hacer**

3: Verificar que la columna sea completa. Una columna es completa si no tiene símbolos vacíos("–")

4: **Si** la columna es completa **Entonces**

5: Verificar que no existan actividades que estén antes y también detrás de la columna, de cumplirse esto *separador* = verdadero. Esto permite desechar las columnas correspondientes a actividades que están en paralelo o que forman un lazo.

6: **Si** *separador* = verdadero **Entonces**

7: Adicionar el valor de la columna a la lista *ListaSeparador*

FinSi

FinSi

FinPara

8: **Si** |*ListaSeparador*| > 0 **Entonces**

9: Formar los bloques de construcción a partir de la lista *ListaSeparador* y almacenarlos en *ListaHijos*

FinSi

10: Devolver *ListaHijos*

Anexo 3 Algoritmo para la búsqueda de lazo

Algoritmo Buscar lazo

Entrada: Bloque de construcción $C^{i}_{\mathcal{A}}$

Salida: Lista de bloques de construcción

1: Crear una lista vacía *ListaHijos*. *ListaHijos* almacena los bloques de construcción obtenidos en una posible descomposición

2: Definir inicio y fin del análisis. Es posible que el bloque de construcción tenga toda la primera columna ocupada por una actividad (no hay símbolos "–") que representa el evento de inicio del subproceso. Esta actividad debe ser eliminada del análisis (el análisis comenzaría en la segunda columna), de manera análoga se procede con la última columna, la cual puede corresponderse con el evento de fin del subproceso. La actividad contenida en la primera columna no puede estar duplicada de lo contrario se considera esta columna como parte del análisis

3: Buscar la primera actividad que aparece en cada fila del bloque de construcción

4: **Si** la primera actividad es la misma en cada fila **Entonces**

5: Definir las secuencias delimitadas por la primera actividad o el fin del bloque de construcción.

6: Alinear las secuencias de actividades definidas. La alineación se realiza con los mismos parámetros definidos en el *Componente para la alineación de las trazas*

7: Construir un bloque de construcción (*BC*) con la matriz obtenida en la alineación

8: Adicionar *BC* a *ListaHijos*

FinSi

9: Devolver *ListaHijos*

Anexo 4 Algoritmo para la búsqueda de XOR-OR

Algoritmo Buscar XOR-OR

Entrada: Bloque de construcción $C^{i}_{\mathcal{A}}$

Salida: Lista de bloques de construcción

1: Crear una lista vacía *ListaHijos*. *ListaHijos* almacena los bloques de construcción obtenidos en una posible descomposición

2: Definir inicio y fin del análisis. Este paso es igual al que se realiza en el algoritmo *Buscar lazo*, paso 2.

3: Construir *n* conjuntos. Donde cada conjunto contiene una fila diferente del bloque de construcción y esta fila representa el elemento raíz del conjunto. La raíz de un conjunto es el número de una fila del bloque de construcción, dicha fila debe estar contenida en el conjunto y lo representa unívocamente. El procedimiento *raíz*(*x*) determina el elemento raíz en el conjunto a donde *x* pertenece, siendo *x* el número de una fila en el bloque de construcción.

4: **Para** *i* = 1 **Hasta** *n* **Con Paso** 1 **Hacer**

5: **Para** *j* = 1 **Hasta** *n* **Con Paso** 1 **Hace**r

6: **Si** *raíz*(*i*) ≠ *raíz*(*j*) **Entonces**

7: **Si** fila *i* y fila *j* tienen actividades en común **Entonces**

8: *Unir*(*raíz*(*i*), *raíz*(*j*)). El procedimiento *Unir*(*x*,*y*) une los conjuntos que contienen a *x* y *y* como raíces, respectivamente.

FinSi

FinSi

FinPara

FinPara

9: **Si** la cantidad de conjuntos > 1 **Entonces**

10: **Para** cada conjunto **Hacer**

11: Crear un bloque de construcción que contiene como filas los elementos del conjunto. En este caso se ha descompuesto el proceso analizado mediante un XOR.

12: Adicionar el bloque de construcción creado a *ListaHijos*

FinPara

Sino

13: **Para** *i* = 1 **Hasta** *n* **Con Paso** 1 **Hacer**

14: Verificar que la fila *i* es base del OR. Una fila es base del OR si no se puede descomponer completamente a partir de la unión de otras filas
15: **Si** la fila *i* es base del OR **Entonces**
16: Crear un conjunto con la fila *i* contenida. La raíz del conjunto es *i*
FinSi
FinPara
17: **Para** cada base del OR *x* **Hacer**
18: **Para** cada base del OR *y* **Hacer**
19: **Si** *x* tiene actividades en común con *y* **Entonces**
20: *Unir*(*raíz*(*x*), *raíz*(*y*)).
FinSi
FinPara
FinPara
21: **Si** la cantidad de conjuntos > 1 **Entonces**
22: **Para** cada conjunto **Hacer**
23: Crear un bloque de construcción que contiene como filas los elementos del conjunto. En este caso se ha descompuesto el proceso analizado mediante un OR.
24: Adicionar el bloque de construcción creado a *ListaHijos*
FinPara
FinSi
25: Devolver *ListaHijos*

Anexo 5 Algoritmo para la búsqueda de paralelismo

Algoritmo Buscar paralelismo

Entrada: Bloque de construcción $C^{i}_{\mathcal{A}}$

Salida: Lista de bloques de construcción

1: Crear una lista vacía *ListaHijos*. *ListaHijos* almacena los bloques de construcción obtenidos en una posible descomposición

2: Definir inicio y fin del análisis. Este paso es igual al que se realiza en el algoritmo *Buscar lazo*, paso 2.

3: Formar *k* conjuntos que cumplen las condiciones siguientes. Entre las actividades que conforman un conjunto no existe una relación de paralelismo. Una relación de paralelismo se establece entre dos actividades cuando en el bloque de construcción existe una fila en la que la actividad *a* está antes que la *b* y existe otra fila en la que la actividad *b* está antes que la *a*. Además, se cumple que para todo *a*, *b*; tal que *a* pertenece al conjunto *A* y *b* pertenece al conjunto *B*, siendo *A* y *B* conjuntos disjuntos de actividades contenidas en el bloque de construcción analizado; *a* está en paralelo con *b*.

4: **Si** $k > 1$ **Entonces**

5: **Para** cada conjunto **Hacer**

6: Crear un bloque de construcción formado a partir de las columnas del bloque analizado que contienen las actividades incluidas en el conjunto.

7: Adicionar el bloque de construcción creado a *ListaHijos*

FinPara

FinSi

8: Devolver *ListaHijos*

Anexo 6 Algoritmo para la búsqueda de secuencia oculta

Algoritmo Buscar secuencia oculta

Entrada: Bloque de construcción $C^i_{\mathcal{A}}$

Salida: Lista de bloques de construcción

1: Crear una lista vacía *ListaHijos*. *ListaHijos* almacena los bloques de construcción obtenidos en una posible descomposición

2: Determinar el orden de precedencia entre las actividades que conforman el bloque de construcción. Una actividad *a* precede a otra actividad *b* si en todas las filas *a* está antes que *b*

3: Determinar grupos de actividades, donde las actividades que conforman un grupo no tienen orden de precedencia

4: Determinar el orden de precedencia entre los grupos. Un grupo *A* precede a otro grupo *B* si todas las actividades de *A* preceden a todas las actividades de *B*

5: Ajustar el bloque de construcción considerando los grupos y la precedencia establecida. Este paso garantiza que existe una coherencia entre la precedencia establecida entre las actividades y el orden de las columnas en el bloque de construcción

6: **Para** i = 1 **Hasta** m **Con Paso** 1 **Hacer**

7: $B_i = \beta(1, i, C^i_{\mathcal{A}})$. Donde B_i es el valor asociado a la evaluación de los bloques de construcción que forman una secuencia oculta desde la primera columna hasta la columna i. $\beta(x, y, BC)$ es un procedimiento que determina el valor de μ para el bloque de construcción formado por las columnas desde la x hasta la y del bloque de construcción BC.

8: **Para** j=2 **Hasta** i **Con Paso** 1 **Hacer**

9: **Si** $B_i > B_{j-1} + \beta(j, i, C^i_{\mathcal{A}})$ **Entonces**

10: $B_i = B_{j-1} + \beta(j, i, C^i_{\mathcal{A}})$

11: $S_i = j$

FinSi

FinPara

FinPara

12: **Si** $B_m \neq \infty$ **Entonces**

13: $p = m$

14: **Mientras** $p > 1$ **Hacer**

15: Adicionar la sub-matriz que se obtiene desde las columnas S_p hasta la p a *ListaHijos*

16: $p = S_p - 1$

FinMientras

FinSi

17: Devolver *ListaHijos*

Anexo 7 Algoritmo para la propagación de información estimada

Algoritmo Propagar información estimada

Entrada: Árbol de bloques de construcción Ĕ

Salida: Árbol de bloques de construcción

1: **Para** cada bloque de construcción contenido en Ĕ **Hacer** //haciendo el recorrido in-orden

2: **Para** cada actividad invisible del bloque de construcción analizado **Hacer**

3: Verificar las referencias de la actividad invisible analizada al bloque de cosntrucción padre, es decir, se identifica la posición en el bloque de construcción padre donde debe insertarse la actividad invisible del bloque de construcción hijo

4: Adicionar en el bloque de construcción padre una columna con la actividad invisible según corresponda y en correspondencia con las posiciones definidas en el paso anterior

5: Desplazar en el bloque de construcción padre las restantes columnas y actualizar la referencia al bloque de construcción padre de este último

FinPara

FinPara

6: Devolver el árbol de bloques de construcción Ĕ modificado

Anexo 8 Algoritmo para la construcción del registro de evento con IE

Algoritmo Construir registro de evento con IE

Entrada: Bloque de construcción $C^1_{\mathcal{A}}$, Registro de evento original L

Salida: Registro de evento

1: Eliminar de $C^1_{\mathcal{A}}$ todos los símbolos vacíos ("–").

2: **Para** cada caso contenido en **Hacer**

3: Buscar las coincidencias del caso analizado en cada caso de L

4: Insertar las actividades invisibles en L según las coincidencias detectadas

FinPara

5: Devolver L modificado

Anexo 9 Modelo del proceso Gestionar recursos

Modelo representativo del proceso Gestionar recursos.