Revista Cubana de Ciencias Informáticas
Vol. 9, No. 4, Mes Octubre-Diciembre, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-215

Tipo de artículo: Artículo original
Temática: Técnicas de programación
Recibido: 29/09/2014 | Aceptado: 01/09/2015

Algoritmo para la identificación de variantes de procesos

Algorithm for variants process identification

Damián Pérez Alfonso[1*], Raykenler Yzquierdo Herrera[1], Eudel Pupo Hernández[1], Reynaldo López Jiménez[1]

[1]Facultad 3. Universidad de las Ciencias Informáticas, Carretera a San Antonio de los Baños, km $2\frac{1}{2}$, Torrens, Boyeros, La Habana, Cuba. CP.: 19370

[*]Autor para correspondencia: dalfonso@uci.cu

Resumen

La minería de proceso es una disciplina que impulsa el desarrollo de técnicas y herramientas para analizar los procesos partiendo de los registros de eventos. Las técnicas de minería de proceso son utilizadas en diferentes etapas de la gestión de procesos de negocio, incluyendo el diagnóstico. El diagnóstico del proceso ayuda a tener una visión general del proceso y de los aspectos más significativos del mismo. Las técnicas de minería de proceso para el diagnóstico son afectadas por el ruido y la ausencia de información en los registros de eventos. Esto dificulta la identificación de los patrones de control de flujo del proceso, limitando la consecución de los objetivos del diagnóstico. En este trabajo se propone un algoritmo para la identificación de variantes de modelos de proceso que considera el ruido y la ausencia de información en la identificación de los patrones de control de flujo. Utilizando una implementación de este algoritmo se experimentó con registros de eventos que combinan situaciones de ruido y ausencia de información. Los resultados obtenidos muestran que el algoritmo identifica correctamente los patrones de control de flujo, aún con la presencia de ruido y ausencia de información.

Palabras claves: algoritmo, minería de proceso, patrones de control de flujo, variantes de proceso

Abstract

Process mining is a discipline that impulse tools and techniques development for process analysis, starting from event logs. Process mining techniques are used in differents stages of business process management, including diagnosis. Process diagnosis is useful to obtain a general process view, and it's more significative elements. Event logs characteristics like noise and lack of information affects process mining techniques in process diagnosis stage. On these scenarios identification of control flow patterns become a rough task, so

Revista Cubana de Ciencias Informáticas
Vol. 9, No. 4, Mes Octubre-Diciembre, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-215

diagnosis objectives can be complicated to achieve. On this work, an algorithm for identification of process models variants is presented. The proposed solution takes into account the noise and lack of information. An experiment was performed with event logs that combine noise and lack of information, using an implementation of the algorithm proposed. Obtained results show that proposed algorithm identifies properly the control flow patterns even on events logs with noise and lack of information.

Keywords: *algorithm, control flow patterns, process mining, process variants*

Introducción

La Gestión de Procesos de Negocio (BPM) está orientada a la identificación y gestión sistemática de los procesos de una organización para el logro eficaz y eficiente de los objetivos de la empresa. Como parte de este modelo de gestión se han desarrollado sistemas de información para controlar la ejecución de los procesos, evaluar su funcionamiento y apoyar la toma de decisiones (Weske, 2007). Estos sistemas suelen poseer registros de eventos, donde almacenan información sobre la ejecución de los procesos que soportan. La minería de proceso es una disciplina dirigida al desarrollo de técnicas y herramientas para analizar los registros de eventos, extraer información a partir de ellos y presentar de forma explícita el conocimiento que contienen (van der Aalst, 2011). El estudio de los registros de eventos facilita el análisis del funcionamiento real de una empresa a partir de sus procesos y el orden en que se realizan (Outmazgin y Soffer, 2014).

El ciclo de vida de BPM abarca siete fases: diagnóstico, (re) diseño, análisis, implementación, (re) configuración, ejecución y ajuste (van der Aalst, 2011). El diagnóstico de proceso comprende el análisis de rendimiento, la detección de anomalías, la identificación de patrones comunes y de desviaciones en el proceso (Bose, 2012). Las técnicas de minería de proceso para el diagnóstico identifican los comportamientos registrados con mayor frecuencia, lo que posibilita dirigir la mejora del proceso hacia los elementos más críticos. También detectan ejecuciones anómalas en el registro de eventos lo cual brinda información acerca de posibles fraudes o violaciones en las políticas (Bose, 2012). Adicionalmente, la identificación y tratamiento del ruido y la ausencia de información, así como la identificación de patrones de control de flujo son relevantes para el diagnóstico, por su repercusión en la comprensión del proceso y de las relaciones entre sus actividades (De Weerdt et al., 2012).

En minería de proceso se considera ruido al comportamiento raro e infrecuente contenido en el registro de eventos y que no es representativo del comportamiento común del proceso. Aunque en ocasiones se ve como el resultado de errores ocurridos al registrar los eventos, no existe una forma explícita de identificar en el registro dichos errores, por lo cual se debe considerar el ruido como desviaciones del proceso (van der Aalst, 2011).

La identificación de patrones de control de flujo permite determinar las actividades que se realizan sincrónicamente, los bloques de actividades que se repiten, el orden en que se ejecutan determinadas actividades y otros comportamientos relevantes en el proceso (Bose, 2012). El tratamiento del comportamiento infrecuente es relevante para los algoritmos que identifican patrones de control de flujo. Descartar el ruido equivale a eliminar comportamiento del proceso y considerarlo puede conducir a modelos poco estructurados y complejos. Los algoritmos se basan en el descubrimiento del comportamiento más común, a partir del número de repeticiones de sucesiones directas de las parejas de eventos, denominada envergadura de evento y del número de trazas idénticas, llamado envergadura de traza (Bratosin, 2011).

Para identificar los patrones de control de flujo es necesario que el registro de eventos contenga información suficiente, es decir, posea un nivel de completitud tal que sus trazas sean representativas del comportamiento el proceso (van der Aalst et al., 2004). La completitud puede ser afectada por la ausencia de información, se denomina de esta forma a la ausencia de evidencia en el registro de eventos de la ejecución de algunas tareas del proceso. Estas tareas invisibles pueden no haber sido registradas debido a errores, a que el sistema de información no deja huella de su ejecución o a que no fueron informatizadas (Yzquierdo-Herrera, 2012).

Las técnicas de diagnóstico Análisis de Diagramas de Puntos (Molka et al., 2013) y Visualización de Flujo y Alcance (Gunther, 2009) no son capaces de identificar patrones de control de flujo y solo muestran una vista general del proceso. Por otra parte, la técnica Tandem Arrays (Bose y van der Aalst, 2009) extrae patrones de ejecución del registro de eventos, pero no obtiene las relaciones o dependencias entre ellos. La Minería Difusa (Gunther y van der Aalst, 2007) obtiene del registros de eventos un mapa del proceso que muestra las actividades y sus relaciones, aunque es robusta ante el ruido, no identifica los patrones de control de flujo. La Descomposición en Subprocesos utilizando bloques de construcción (Yzquierdo-Herrera et al., 2013) sí identifica los patrones de control de flujo presentes, considerando además la ausencia de información. Sin embargo, la identificación de los patrones se ve afectada por la presencia de ruido en los registros de eventos.

La detección de patrones de control de flujo, identificación y tratamiento del ruido y ausencia de información en los registros de eventos no son consideradas integralmente por las técnicas de diagnóstico de proceso. Esto se traduce en modelos complejos y desviados de la ejecución real, identificación errónea de los patrones de ejecución y contextualización incorrecta de las anomalías. Por tanto, debido a las limitaciones de las técnicas analizadas se dificulta la comprensión de la estructura del proceso, así como del comportamiento y las anomalías presentes en el registro de eventos, comprometiendo el cumplimiento de los objetivos del diagnóstico.

El algoritmo propuesto en el presente trabajo tiene como objetivo identificar los patrones de control de flujo considerando el ruido y la ausencia de información presente en el registro de eventos. En lugar de construir un único modelo a partir del registro de eventos se construye un árbol de variantes de modelos del proceso.

Revista Cubana de Ciencias Informáticas
Vol. 9, No. 4, Mes Octubre-Diciembre, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-215

Materiales y métodos

El algoritmo desarrollado utiliza como entrada un registro de eventos en formato XES[1] cuyo contenido debe ser coherente con las definiciones 1 y 2 (tomadas de (van der Aalst, 2011)). Los patrones de control de flujo que se presentan en la Definición 3 se identifican a partir del comportamiento registrado (Definición 4).

Definición 1 Eventos*: Sea $\mathscr{E}$ el universo de eventos. Los eventos pueden caracterizarse por varios atributos. Siendo AN el conjunto de nombres de atributos, para cualquier evento $e \in \mathscr{E}$ y un nombre $n \in AN : \#_n(e)$ es el valor del atributo n para el evento e. $\mathscr{E}^*$ representa el conjunto de todas las secuencias finitas sobre $\mathscr{E}$. Una secuencia finita sobre $\mathscr{E}$ de longitud n es un mapeo $\sigma \in 1, ..., n \to \mathscr{E}$.*

Definición 2 Traza, registro de eventos: *Sea $\mathscr{C}$ el universo de casos. Para cualquier caso $c \in \mathscr{C}$ y un nombre $n \in AN : \#_n(c)$ es el valor del atributo n para el caso c. Cada caso tiene un atributo obligatorio trace $\#_{trace}(c) \in \mathscr{E}^*$. $\hat{c} = \#_{trace}(c)$ es un atajo para referirse a la traza de un caso.*

Una traza es una secuencia finita de eventos $\sigma \in \mathscr{E}^*$ tal que cada evento aparece una sola vez, o sea, para $1 \leq i \leq j \leq |\sigma| : \sigma(i) \neq \sigma(j)$.

Un registro de eventos es un conjunto de casos $L \subseteq \mathscr{C}$ tal que cada evento aparece al menos una vez. Si el registro de eventos contiene marcas de tiempo, los eventos deben ser ordenados dentro de una traza según esas marcas: $\forall c \in L, i$ y j tal que $1 \leq i < j|\hat{c}| : \#_{time}(\hat{c}(i)) \leq \#_{time}(\hat{c}(j))$.

Definición 3 Patrones de control de flujo:

- Secuencia: *dos subprocesos se ejecutan secuencialmente si uno ocurre inmediatamente después del otro.*
- Selección exclusiva: *dos subprocesos forman parte de una selección exclusiva si, en un punto de decisión, se puede ejecutar solamente uno de los dos.*
- Selección no exclusiva: *dos subprocesos son opciones de una selección no exclusiva si, en un punto de decisión, pueden ejecutarse ambos o solamente uno de ellos.*
- Paralelismo: *dos subprocesos se ejecutan en paralelo si ambos se ejecutan simultáneamente.*
- Lazo: *dos subprocesos se encuentran en un lazo si se pueden repetir múltiples veces. Cada repetición comienza con la ejecución del primer subproceso (**Do**), continúa con el segundo (**Redo**) y termina con el **Do**. El **Redo** puede ser un subproceso vacío, por lo que en este caso el único repetido sería el **Do**.*

[1] eXtensible Event Stream: www.xes-standard.org

Revista Cubana de Ciencias Informáticas
Vol. 9, No. 4, Mes Octubre-Diciembre, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-215

Definición 4 Comportamiento del proceso: *Sea P un proceso y S el conjunto de todos los subprocesos de P. L denota un registro de eventos generados por P y A el conjunto de actividades de P. Sea $X_i \subseteq A$ el conjunto de actividades de un subproceso $s_i \in S$ para $1 \leq i \leq |S|$. Una sección del registro de eventos (sublog) l_i es el multiconjunto conformado por todas las proyecciones no vacías de trazas $\sigma \in L$ sobre X_i.*

Una sucesión directa entre a y b se denota como $a >_{l_i} b$ y expresa la existencia de al menos una traza $\sigma = \langle t_1, t_2, ..., t_n \rangle$ donde $1 \leq i < n$ tal que $\sigma \in l_i$ y $t_i = a$ y $t_{i+1} = b$.

Una sucesión indirecta entre a y b se denotada por $a \gg_L b$ y expresa la existencia de al menos una traza $\sigma = \langle t_1, t_2, ..., t_n \rangle$ donde $1 \leq i < j \leq n$ tal que $\sigma \in L$ y $t_i = a$ y $t_j = b$.

Siendo $>_{l_i}$ el multiconjunto de sucesiones directas presentes en l_i y $\gg_{l_i}$ el multiconjunto de sucesiones indirectas presentes en l_i. El comportamiento de s_i observable en l_i y denotado por β_i está conformado por $>_{l_i}$, $\gg_{l_i}$ y los multiconjuntos de eventos que inician trazas ($\triangleright_{l_i}$), terminan trazas ($\not\triangleright_{l_i}$) y se encuentran repetidos en la misma traza ($\trianglerighteq_{l_i}$).

El algoritmo busca varias descomposiciones alternativas para el mismo subproceso, utilizando diferentes patrones de control de flujo. Las alternativas se construyen, descartando o no, determinados comportamientos presentes en el registro de eventos. También se pueden construir, considerando o no, determinados comportamientos ausentes. Los comportamientos que son descartados como ruido y los que son asumidos como ausentes en una variante, no lo son en otra. Esto permite controlar el impacto estructural del ruido y la ausencia de información en la construcción de las alternativas. Las diferentes alternativas de descomposición que pueden existir en cada subproceso conforman variantes del proceso, concepción expuesta en la Definición 5.

Definición 5 Variantes de proceso: *Las variantes de un modelo de procesos o variantes de proceso, son modelos de un proceso que describen el mismo proceso de negocio, y poseen algunas diferencias estructurales. Las diferencias están dadas por los patrones de control de flujo que se utilizan en secciones equivalentes del proceso y la presencia de determinadas actividades.*

Partiendo de las descomposiciones alternativas por cada subproceso, el algoritmo construye un modelo jerárquico que representa las diferentes variantes de modelos del proceso. Este modelo de proceso denominado *Árbol de Variantes* se presenta como parte de esta investigación y se describe en la Definición 6. En el *Árbol de Variantes* se utiliza un operador para representar cada patrón de control de flujo de la Definición 3: el operador $\rightarrow$ para el patrón *Secuencia*, $\wedge$ para *Paralelismo*, $\times$ para *Selección exclusiva*, $\vee$ para *Selección exclusiva* y $\circlearrowleft$ para el patrón *Lazo*.

Revista Cubana de Ciencias Informáticas
Vol. 9, No. 4, Mes Octubre-Diciembre, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-215

DEFINICIÓN 6 *Un* Árbol de Variantes *(VT) es una representación de diferentes descomposiciones en subprocesos aplicadas a un proceso P a partir de un registro de eventos L generado por la ejecución de P. Denotemos por S el conjunto de subprocesos representado en VT, por A el conjunto de actividades que conforman P, por $\sim$ la actividad invisible ($\sim \notin A$) y por $W = \{\rightarrow, \wedge, \times, \vee, \circlearrowleft\}$ el conjunto de operadores. Un VT es un árbol en el cual:*

- *Cada nodo es un nodo subproceso o un nodo operador.*
- *Un nodo subproceso $sn_i; 0 < i \leq |S|$ representa un subproceso $s_i \in S$.*
- *Un nodo operador $pn_i^w; 0 < i \leq |S|$, representa la descomposición de s_i mediante un operador $w \in W$.*
- *El nodo raíz es un nodo subproceso relativo a P.*
- *Un nodo operador tiene n nodos subproceso como hijos con $2 \leq n < |S|$.*
- *Un nodo subproceso tiene p nodos operador como hijos con $0 \leq p \leq |W|$.*
- *Cada nodo hoja es un nodo subproceso relativo a una actividad $a \in A \cup \{\sim\}$.*

El *Árbol de Variantes* de la Figura 1 representa el comportamiento contenido en un registro de eventos $L = [\langle b, b, a, e\rangle^1, \langle b, b, b, a, e, d\rangle^4, \langle b, a, b, d, e\rangle^5, \langle a, e, c\rangle^3, \langle b, b, c, e\rangle^2]$ [2] generado por un proceso conformado por los subprocesos del conjunto $S = \{s_1, s_2, s_3, s_4, s_5\}$. Como puede apreciarse, existen diferentes nodos operador para el mismo subproceso, los cuales pueden tener diferentes subárboles (s_2) o no (s_3).

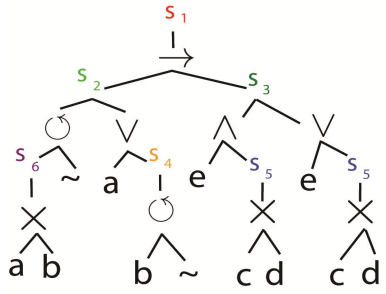

Figura 1. *Árbol de Variantes* construido a partir de L

[2]Los exponentes denotan la frecuencia de aparición de cada traza en el registro de eventos.

Revista Cubana de Ciencias Informáticas
Vol. 9, No. 4, Mes Octubre-Diciembre, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-215

Algoritmo para la identificación de variantes de proceso

El algoritmo que se propone para identificar las variantes del proceso, a partir del comportamiento contenido en un registro de eventos, está compuesto por dos procedimientos que se realizan iterativamente, la descomposición en subprocesos (Algoritmo 1) y la búsqueda de variantes de descomposición a partir de la identificación de patrones de control de flujo (Algoritmo 2). Además, el segundo procedimiento incluye un algoritmo para determinar los estados vecinos en la búsqueda (Algoritmo 3).

El algoritmo de descomposición en subprocesos (Algoritmo 1) utiliza un registro de eventos L generado por un proceso P y umbrales de ruido y completitud (T) para construir un *Árbol de Variantes* VT. Siendo S el conjunto de todos los subprocesos del proceso P, para cada nuevo subproceso $s_i \in S$, se busca una variante de descomposición según cada uno de los patrones de control de flujo.

Algoritmo 1 Descomposición en subprocesos

Entrada: registro de eventos (L), umbrales de ruido y completitud (T)
Salida: *Árbol Variantes* (VT)

Procedimiento DECOMPOSE(L, T)
  $W \leftarrow \{\rightarrow, \wedge, \times, \vee, \circlearrowleft\}$ ▷ Operadores de control de flujo para la descomposición
  $sn_1 \leftarrow$ CREATESUBPROCESS(L) ▷ Crear un nodo subproceso relativo a P
  $VT \leftarrow sn_1$ ▷ Asignar el nodo sn_1 como raíz de VT
  $S' \leftarrow \{sn_1\}$ ▷ Crear una cola para los subprocesos pendientes de análisis.
  while $|S'| \geq 1$ **do**
    $sn_i \leftarrow$ GETFIRST(S') ▷ Extraer el primer nodo subproceso en S'
    if s_i contiene más de una actividad **then**
      $\beta_i \leftarrow$ EXTRACTBEHAVIOR(sn_i) ▷ Extraer el comportamiento contenido en l_i
      for all w en W **do**
        $V \leftarrow$ FINDVARIANTS(w, β_i, T) ▷ Algoritmo 2
        if $|V| > 0$ **then** ▷ Si s_i puede ser descompuesto por w dentro de T
          $pn_i^w \leftarrow$ CREATEOPERATORNODE(w, V) ▷ Crear un nodo operador para el VT
          for all nodo subproceso k en V **do**
            adicionar el nodo subproceso k como hijo de pn_i^w ▷ Modificar VT
            adicionar el nodo subproceso k a S'
          end for
          adicionar el nodo pn_i^w como hijo del nodo sn_i ▷ Modificar VT
        end if
      end for
    end if
  end while
  return VT, D
end Procedimiento

Los subprocesos generados por cada descomposición son añadidos a S' para ser analizados posteriormente, solo si incluyen más de una actividad. El *Árbol de Variantes* (VT) se va construyendo a medida que se van identificando las variantes, al añadir el nodo operador como hijo del subproceso al que corresponde la variante y los nuevos nodos subprocesos, resultantes de la descomposición, como hijos del nodo operador.

Los umbrales definen con un número entre cero y uno la cantidad de comportamiento dentro de un subproceso que puede ser descartado o asumido como ausente para descomponer un subproceso considerando determinado patrón de control de flujo. Por ejemplo, un umbral de 0,2 para ruido en el patrón secuencia significa que hasta el 20 % del comportamiento expresado en el registro de eventos puede ser descartado para encontrar una descomposición mediante secuencia. De manera similar, un 0,8 de completitud para paralelismo indica que el subproceso puede ser descompuesto mediante ese patrón conteniendo solamente el 80 % del comportamiento necesario para ser identificado, ya que el 20 % restante se asume como ausente.

La existencia de variantes de descomposición depende del comportamiento registrado por cada subproceso (β_i). Se denota con ϑ_i^w al comportamiento del proceso en s_i para el patrón que representa w y que no forma parte de β_i. De manera similar se denota con $\beta_i^w \subseteq \beta_i$ al comportamiento en β_i que se ajusta a la descomposición de s_i mediante w. Se puede encontrar una variante de descomposición para s_i por un operador w siempre que $\beta_i \setminus \beta_i^w$ pueda ser descartado dentro del umbral de ruido y ϑ_i^w pueda ser asumido según el umbral de completitud.

Para encontrar una variante de descomposición en un subproceso $s_i \in S$ es necesario identificar un patrón de control de flujo, lo cual equivale a encontrar varios subprocesos $s_k | \forall k \neq i : s_k \in S$, que cumplan con lo establecido en la Definición 3. En términos de Teoría de Conjuntos esto es igual a encontrar una partición de X_i (el conjunto de actividades que forman s_i). En principio pueden existir tantas descomposiciones por un patrón de control de flujo como particiones de X_i. Encontrar los s_k es equivalente a encontrar las actividades que pertenecen a cada uno. Como una actividad puede pertenecer únicamente a un subproceso esto significa encontrar n conjuntos disjuntos de las actividades en X_i. De todas las particiones posibles por cada patrón de control de flujo solamente interesa la que mejor se ajusta a β_i, o sea, aquella que requiere descartar menos comportamiento de β_i y asumir menos comportamiento ausente de β_i para que los subprocesos s_k cumplan con lo establecido en la Definición 3.

La búsqueda de la mejor variante de descomposición para s_i por cada patrón se realiza mediante el procedimiento que se expone en el Algoritmo 2 y está basada en la **Búsqueda de costo uniforme**, la cual expande el nodo con menor costo de camino. Si el costo de cada paso es mayor o igual que una pequeña constante positiva ε, se garantiza que el método es completo y óptimo. La complejidad temporal y espacial del peor caso de este método de búsqueda pueden ser descritas por la ecuación $r^{1+C/\varepsilon}$, donde r es el factor de ramificación del árbol y C es el costo del camino de la solución óptima (Russell et al., 1995).

Revista Cubana de Ciencias Informáticas
Vol. 9, No. 4, Mes Octubre-Diciembre, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-215

Algoritmo 2 Búsqueda de variantes

Entrada: operador (w), comportamiento en $l_i(\beta_i)$, umbrales de ruido y completitud (T)
Salida: variante de descomposición en subprocesos a través del operador w

Procedimiento FINDVARIANTS(w, β_i, T)
  $\beta'_i \leftarrow$ SORTBYFREQUENCY(β_i) ▷ Ordenar el comportamiento descendentemente por su frecuencia
  $\tau_w \leftarrow T[w]$ ▷ Obtener umbrales para el operador w
  $\Upsilon_i^w \leftarrow$ CREATEFIRSTDECOMPOSITIONPROPOSAL(w, β'_i, τ_w)
  if $\Upsilon_i^w = \emptyset$ **then** ▷ β'_i no es suficiente para crear una descomposición bajo τ_w
    return $\emptyset$
  end if
  $\psi_1 \leftarrow$CREATESEARCHNODE(β'_i, Υ_i^w)
  $\Psi \leftarrow \{\psi_1\}$ ▷ Crear cola de nodos abiertos, con prioridad según el costo del camino
  while $\psi_n \leftarrow$ extraer primer nodo en Ψ **do**
    if ψ_n es "meta" & $\beta'_i = \emptyset$ **then**
      $g \leftarrow \psi_n$
    else if ψ_n no ha sido visitado **then**
      $\Psi_n \leftarrow$ SEARCHNEIGHBORS(w, ψ_n, β'_i, τ_w) ▷ Algoritmo 3
      $\Psi \leftarrow \Psi + \Psi_n$ ▷ Adicionar vecinos como nodos sin visitar
    end if
  end while
  if g no es nulo **then** ▷ Si se encontró una variante de descomposición
    $\Upsilon_i^w \leftarrow$ descomposición potencial en g
    $SP \leftarrow$ CREATENEWSUBPROCESSES(w, Υ_i^w, β_i) ▷ Crear los subprocesos
    return CREATEVARIANT(SP, g, β_i) ▷ Crear la variante identificada
  else
    return $\emptyset$
  end if
end Procedimiento

En la búsqueda de variantes (Algoritmo 2) cada nodo en el espacio de búsqueda (ψ_n) está definido por Υ_i^w, el comportamiento no procesado ($\beta'_i \subseteq \beta_i$), el comportamiento descartado (η_i^w) y el comportamiento asumido (θ_i^w). Al adicionar nuevos nodos (Ψ_n), en caso de existir nodos con la misma descomposición (Υ_i^w), solamente permanece aquel con menor cantidad de comportamiento por procesar (β'_i) independientemente del costo de camino. El objetivo se alcanza cuando β'_i ha sido procesado y Υ_i^w posee al menos dos conjuntos disjuntos.

La creación de nuevos nodos de búsqueda, descrita en el Algoritmo 3, se realiza utilizando dos operadores. Si partiendo de Υ_i^w al procesar un comportamiento $b_x \in \beta'_i$ se genera una descomposición diferente ($\Upsilon_i^{w'}$) se comprueba si b_x puede descartarse y se genera un nuevo nodo. Si existe un comportamiento $b_y \in \vartheta_i^w$ (comportamiento de s_i según el patrón que representa w y que no forma parte de β_i) y este puede ser asumido con el objetivo de mantener Υ_i^w también se genera un nuevo nodo.

Revista Cubana de Ciencias Informáticas
Vol. 9, No. 4, Mes Octubre-Diciembre, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-215

Algoritmo 3 Buscar vecinos

Entrada: operador (w), nodo actual (ψ_0), comportamiento (β'_i), umbrales para w (τ_w)
Salida: nodos vecinos que representan los estados alcanzables desde ψ_0

Procedimiento searchNeighbors(w, ψ_0, β'_i, τ_w)
  $\Upsilon_i^w \leftarrow$ extraer descomposición en ψ_0
  $\Psi_n \leftarrow \{\}$ ▷ Crear el conjunto de nodos vecinos
  while $|\beta'_i| > 0$ & $|\Upsilon_i^w| > 1$ **do**
    $b_x \leftarrow \beta'_i$
    $\beta'_i \leftarrow \beta'_i \setminus b_x$
    $\Upsilon_i^{w'} \leftarrow$ processBehaviorByPattern(Υ_i^w, b_x)
    if $\Upsilon_i^{w'} \neq \Upsilon_i^w$ **then** ▷ Si $\Upsilon_i^{w'}$ es nuevo
      if $(c(b_x) + \psi_0 \rightarrow td) \leq \tau_w$ **then** ▷ Si b_x puede ser descartado
        $\psi_2 \leftarrow$ createSearchNode($\psi_0, \beta'_i, \Upsilon_i^w$)
        $(\psi_2 \rightarrow td) \leftarrow (\psi_0 \rightarrow td + c(b_x))$ ▷ Asignar costo total del camino
        $(\psi_2 \rightarrow \eta_i^w) \leftarrow b_x$ ▷ Descartar b_x en ψ_2
        $\Psi_n \leftarrow \Psi_n + \psi_2$ ▷ Adicionar vecino considerando ruido
      end if
      $b_y \leftarrow$processBehaviorTakingOnIncompleteness(w, b_x, Υ_i^w)
      if b_y no es nulo & $(c(b_y) + \psi_0 \rightarrow td) \leq \tau_w$ **then** ▷ Si puede asumirse b_y
        $\psi_3 \leftarrow$ createSearchNode($\psi_0, \beta'_i, \Upsilon_i^w$)
        $(\psi_3 \rightarrow td) \leftarrow (\psi_0 \rightarrow td + c(b_y))$ ▷ Asignar costo total del camino
        $(\psi_3 \rightarrow \theta_i^w) \leftarrow b_y$ ▷ Asumir b_y en ψ_3
        $\Psi_n \leftarrow \Psi_n + \psi_3$ ▷ Adicionar vecino considerando ausencia de información
      end if
      $\Upsilon_i^w \leftarrow \Upsilon_i^{w'}$ ▷ Cambiar la descomposición de referencia
    end if
  end while
  if $\beta'_i = \emptyset$ & $|\Upsilon_i^w| > 1$ **then** ▷ Si se procesó todo el comportamiento y Υ_i^w es válida
    $\psi_1 \leftarrow$ createSearchNode($\psi_0, \beta'_i, \Upsilon_i^{w'}$)
    $(\psi_1 \rightarrow td) \leftarrow (\psi_0 \rightarrow td)$ ▷ Mantener el costo del camino
    $\Psi_n \leftarrow \psi_1$ ▷ Adicionar vecino "meta"
  end if
  return Ψ_n ▷ Devolver los nodos vecinos del nodo actual (ψ_0)
end Procedimiento

El costo del camino desde un nodo a alguno de sus vecinos es: $c(b_x) = \frac{\varphi(b_x)}{\sum_{n=1}^{|\beta_i|} \varphi(b_n)}$, si $b_x \in \beta'_i$ es descartado y $c(b_y) = \frac{\varphi(b_y)}{\sum_{n=1}^{|\vartheta_i^w|} \varphi(b_n)}$, si $b_y \in \vartheta_i^w$ es asumido. En ambos casos solo se crea el nuevo nodo si el costo del camino del nodo origen (td) más el costo del camino al nuevo nodo está dentro de los umbrales de ruido y completitud.

Revista Cubana de Ciencias Informáticas
Vol. 9, No. 4, Mes Octubre-Diciembre, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-215

En cada nodo de búsqueda se procesa β_i, según el patrón de control de flujo para el cual se busca una descomposición de s_i (procedimientos processBehaviorTakingOnIncompleteness y processBehaviorByPattern). A continuación se exponen los principios básicos de este procesamiento para cada uno de los patrones.

Secuencia: Existe una descomposición utilizando el operador relativo al patrón secuencia $\Upsilon_i^{\rightarrow} = \{\upsilon_1, ..., \upsilon_m\}$ si $\forall 1 \leq j \leq m$ se cumple que para cualquier sucesión directa del tipo $a >_{l_i} b$ donde $a \in \upsilon_j$ entonces $b \in \upsilon_j$ o $b \in \upsilon_{j+1}$ y además no existe una sucesión indirecta $b \gg_{l_i} a$ tal que $b \in \upsilon_{j+1}$ y $a \in \upsilon_j$.

La primera descomposición potencial es $\Upsilon_i^{\rightarrow} = \{\{a\}, \{b\}\}$ donde $a >_{l_i} b$ es la sucesión directa más frecuente en l_i. A partir de esta descomposición se procesan el resto de las sucesiones en $>_{l_i}$ para adicionar las actividades restantes a los conjuntos que ya existen en $\Upsilon_i^{\rightarrow}$ o a nuevos conjuntos, considerando las reglas antes descritas. Una vez que todas las actividades de X_i están en $\Upsilon_i^{\rightarrow}$, se procesan los eventos que inician ($\triangleright_{l_i}$) y terminan trazas ($\not\triangleright_{l_i}$). Se verifica que si $\Upsilon_i^{\rightarrow} = \{\upsilon_1, ..., \upsilon_m\}$ todas las actividades a que hacen referencia los eventos en $\triangleright_{l_i}$ se encuentren en υ_1 y las actividades cuyos eventos pertenecen a $\not\triangleright_{l_i}$ pertenezcan a υ_m. Si no se cumple esta condición se considera falta de completitud en l_i. Si el evento para el cual no se cumple la condición pertenece a $\triangleright_{l_i}$ se agrega una actividad invisible ($\sim$) a υ_1. Si el evento pertenece a $\not\triangleright_{l_i}$ entonces $\sim$ se agrega a υ_m.

Lazo: Existe una descomposición utilizando el operador relativo al patrón lazo $\Upsilon_i^{\circlearrowleft} = \{\upsilon_1, \upsilon_2\}$ (υ_1 se refiere al subproceso **Do** y υ_2 al subproceso **Redo**) si se cumplen tres condiciones. Primeramente todas las actividades cuyos eventos pertenecen a $\triangleright_{l_i}$ y $\not\triangleright_{l_i}$ forman parte de υ_1 y deben poseer eventos en $\trianglerighteq_{l_i}$. Además, para toda sucesión directa $a >_{l_i} b$ en $>_{l_i}$ donde $a \in \upsilon_1$ y $b \in \upsilon_2$, a debe iniciar al menos una traza. Y por último, para toda sucesión directa $a >_{l_i} b$ en $>_{l_i}$ donde $a \in \upsilon_2$ y $b \in \upsilon_1$, b debe concluir al menos una traza.

En la primera descomposición posible $\Upsilon_i^{\circlearrowleft} = \{\upsilon_1, \upsilon_2\}$, υ_1 contiene las actividades cuyos eventos pertenecen a $\triangleright_{l_i}$ y a $\not\triangleright_{l_i}$ y tienen una frecuencia superior al umbral de ruido. Durante el procesamiento de las sucesiones en $>_{l_i}$ el resto de las actividades son adicionadas a υ_1 o υ_2, considerando las reglas descritas. Se pueden descartar sucesiones indirectas y eventos que inician y terminan trazas, siempre dentro del umbral de ruido.

Paralelismo: Existe una descomposición según el patrón paralelismo $\Upsilon_i^{\wedge} = \{\upsilon_1, ..., \upsilon_m\}$ si $\forall 1 \leq j \leq m$, $\forall 1 \leq k \leq m$ y $j \neq k$ si cumplen dos condiciones. La primera condición es que $\forall a \in \upsilon_j$ y $\forall b \in \upsilon_k$ exista $a \gg_{l_i} b$ y $b \gg_{l_i} a$. Además, en cada una de las trazas debe aparecer al menos un evento de los subprocesos concurrentes, o sea, $\exists a \in \upsilon_j$ en cada traza y $\exists b \in \upsilon_k$ en cada traza.

La primera $\Upsilon_i^{\wedge}$ contiene un conjunto por actividad. Al procesar las sucesiones indirectas ($\gg_{l_i}$) que representan violaciones de las condiciones planteadas se unen los conjuntos en los que se encuentran las actividades de la sucesión procesada. La ausencia de los subprocesos en algunas trazas puede ser ignorada dentro del umbral de

Revista Cubana de Ciencias Informáticas
Vol. 9, No. 4, Mes Octubre-Diciembre, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-215

ruido. Una $\Upsilon_i^{\wedge}$ puede mantenerse asumiendo una sucesión indirecta $b \gg_{l_i} a$ dentro del umbral de completitud.

Selección Exclusiva: Es posible identificar una descomposición $\Upsilon_i^{\times} = \{\upsilon_1, ..., \upsilon_m\}$ si $\forall 1 \leq j \leq m, \forall 1 \leq k \leq m$ y $j \neq k$ se cumple que $\forall a \in \upsilon_j$ y $\forall b \in \upsilon_k$, $\nexists a \gg_{l_i} b$ y $\nexists b \gg_{l_i} a$. La primera descomposición $\Upsilon_i^{\times}$ contiene un conjunto por cada actividad. Al procesar las sucesiones en $\gg_{l_i}$ para las cuales se violan las condiciones planteadas se unen los conjuntos cuyas actividades están asociadas a la sucesión procesada. Una descomposición puede mantenerse si el umbral de ruido permite descartar la sucesión que viola las condiciones planteadas,

Selección no Exclusiva: Existe una variante de descomposición $\Upsilon_i^{\vee} = \{\upsilon_1, ..., \upsilon_m\}$ si $\forall 1 \leq j \leq m, \forall 1 \leq k \leq m$ y $j \neq k$ se cumple que $\forall a \in \upsilon_j$ y $\forall b \in \upsilon_k$ existe $a \gg_{l_i} b$ y $b \gg_{l_i} a$. Además debe comprobarse que no existe representación de todos los subprocesos en al menos una traza. La primera $\Upsilon_i^{\vee}$ contiene un conjunto por cada actividad. El procesamiento de las sucesiones en $\gg_{l_i}$ en las cuales se violan las condiciones planteadas provoca la unión de los conjuntos en los que se encuentran las actividades asociadas a la $a \gg_{l_i} b$ procesada. Una descomposición puede mantenerse asumiendo una sucesión $b \gg_{l_i} a$ dentro del umbral de completitud.

Resultados y discusión

El algoritmo propuesto ha sido implementado en un complemento del marco de trabajo ProM (Verbeek et al., 2011). Los umbrales de ruido y completitud son configurables al iniciar el complemento y en este experimento se utilizaron los valores por defecto: 0,2 para ruido y 0,8 para completitud.

Para comprobar si el algoritmo identifica correctamente los patrones de control de flujo se concibió un experimento con dos grupos de registros de eventos artificiales. Los registros de eventos se generaron a partir de 10 modelos de procesos que combinan diferentes patrones. Los modelos fueron creados con la herramienta *Process Log Generator* (Burattin y Sperduti, 2011), combinando aleatoriamente características como: cantidad de patrones anidados, probabilidad de *Secuencia*, *Lazo*, *AND Split/Join* y *XOR Split/Join*. Los modelos de procesos construidos poseen las características descritas en la Tabla 1.

Por cada uno de estos modelos de proceso se generó un registro de eventos, los cuales fueron utilizados para conformar el grupo G_1. Luego se extrajo a cada registro de eventos perteneciente a G_1 el 5 % de sus eventos, con el objetivo de introducir situaciones de ruido y ausencia de información, conformándose así el grupo G_2. En la Tabla 2 se muestran las características de los registros de eventos de cada grupo.

El diseño experimental propuesto se resume en la Tabla 3, utilizando la siguiente simbología: **G:** Grupo de participantes. Cada grupo está formado por 10 registros de eventos generados aleatoriamente.

Revista Cubana de Ciencias Informáticas
Vol. 9, No. 4, Mes Octubre-Diciembre, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-215

Tabla 1. Características de los modelos de proceso construidos.

Características	**Modelos**									
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Actividades	10	8	11	10	15	10	10	6	13	6
Secuencia	7	6	7	7	9	6	7	6	10	6
Lazo	0	0	0	1	0	1	0	0	0	0
Paralelismo	1	1	2	1	1	0	0	1	1	1
Selección exclusiva	1	0	0	0	2	1	2	0	0	0
Total de patrones	9	7	9	9	12	8	9	7	11	7

Tabla 2. Características de los registros de eventos de cada grupo.

Registro de eventos	Trazas	Grupo G_1		Grupo G_2	
		Eventos	Trazas únicas	Eventos	Trazas únicas
1	1500	10492	3	9961	62
2	1500	12000	2	11399	59
3	1500	16500	12	15675	231
4	1500	23096	90	21945	425
5	5000	35192	9	33473	182
6	1500	18849	38	17466	519
7	5000	32453	3	30849	82
8	1500	9000	2	8552	31
9	5000	65000	6	61762	430
10	5000	30000	2	28563	39

R: Asignación al azar.

X: Estímulo. X_1 corresponde con la extracción de eventos de G_1 y X_2 a la aplicación del algoritmo propuesto.

O: Observación.

En el primer momento las observaciones O_1 y O_3 representan la cantidad de patrones de control de flujo presentes en el modelo original. En el segundo momento las observaciones O_2 y O_4 están asociadas a la cantidad de patrones correctamente identificados por el algoritmo propuesto, con respecto al modelo original. Para cada observación se hacen 10 mediciones, una por cada modelo de proceso. En la Figura 2 se muestra el total de patrones presentes en cada modelo original y los identificados por el algoritmo propuesto, en los registros de eventos de los grupos G_1 y G_2. Se pueden apreciar diferencias para los modelos 1, 5, 6 y 7.

Revista Cubana de Ciencias Informáticas
Vol. 9, No. 4, Mes Octubre-Diciembre, 2015
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 199-215

Tabla 3. Diseño experimental propuesto.

Modelo original			Árbol de Variantes	
R G_1	O_1		X_2	O_2
R G_2	O_3	X_1	X_2	O_4

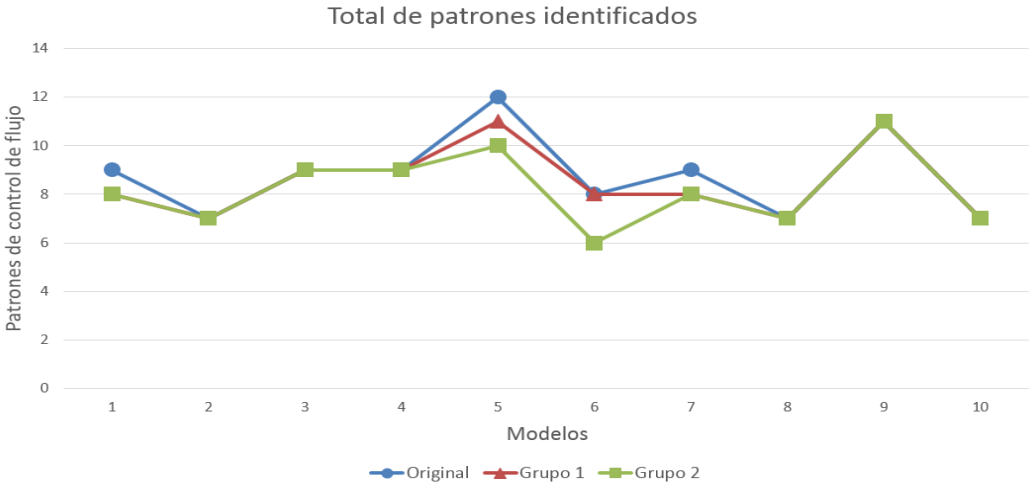

Figura 2. Patrones identificados en el experimento.

Los patrones incorrectamente identificados a partir de los registros de eventos de G_1, respecto a los modelos originales 1, 5 y 7 son en todos los casos *Selección exclusiva*. Las diferencias de los *Árboles de Variantes* obtenidos con G_2, respecto a los obtenidos con G_1 son en el patrón *Selección exclusiva* para 5 y 6, además de en el patrón *Lazo* para el caso 6.

A partir del diseño experimental expuesto se realizaron las pruebas estadísticas. Primeramente se realizaron comparaciones por pares en un grupo, utilizando el test no paramétrico de signos con rasgos de Wilcoxon. Al analizar los datos correspondientes al primer y segundo momento se detectó que no existen diferencias significativas (significación 0.083 para G_1 y 0.063 para G_2) entre los patrones de los modelos originales y los observados en el *Árbol de Variantes*. Para comprobar el impacto del ruido y la ausencia de información en el método propuesto se compararon los valores de las evaluaciones O_2 y O_4 utilizando el test de Mann-Whitney. En esta prueba no se encontraron diferencias significativas entre ambos momentos (significación 0.669).

Las diferencias encontradas entre los *Árboles de Variantes* indican que debe mejorarse el enfoque utilizado para

identificar los patrones *Selección exclusiva* y *Lazo*. Sin embargo, las pruebas estadísticas aplicadas demuestran que estas diferencias no son significativas. Considerando los resultados obtenidos, se puede afirmar que el algoritmo propuesto identifica correctamente los patrones de control de flujo a partir de registros de eventos con ruido y ausencia de información.

Conclusiones

Para resolver la afectación que provocan el ruido y la ausencia de información en la comprensión del proceso es necesario controlar el impacto estructural de estas características. Sin embargo, considerar ciertos comportamientos infrecuentes como ruido y asumir que faltan determinadas evidencias de la ejecución del proceso, son estimaciones que sólo pueden ser confirmadas a partir del contexto de ejecución particular del proceso analizado. Debido a esto el enfoque adoptado por el algoritmo propuesto es buscar varias descomposiciones alternativas para el mismo subproceso, de tal manera que los comportamientos que son descartados como ruido y los que son asumidos como ausentes en una variante de descomposición, no lo son en otra.

Las variantes de descomposición identificadas en el algoritmo forman variantes de modelos del mismo proceso, que son agrupadas en un modelo de proceso jerárquico, el *Árbol de Variantes*. A partir de este árbol el analista del proceso puede escoger entre las variantes construidas, lo cual equivale a decidir cuáles comportamientos deben ser considerados como ruido y descartados del modelo resultante, así como seleccionar cuáles comportamientos están ausentes y cómo deben ser insertados en el modelo.

Las pruebas estadísticas realizadas evidencian que la propuesta identifica correctamente los patrones de control de flujo, en registros de eventos con ruido y/o ausencia de información. El presente trabajo puede extenderse construyendo un algoritmo para identificar, de las variantes obtenidas por el algoritmo propuesto, aquellas que optimicen aspectos de interés para el análisis, como rendimiento temporal o consumo de recursos.

Agradecimientos

Agradecer al Ing. Osiel Fundora Ramírez por su colaboración en la generación de modelos y registros de eventos artificiales.

Referencias

- BOSE, R. (2012). *Process Mining in the Large: Preprocessing, Discovery, and Diagnostics*. PhD thesis, Eindhoven University of Technology.

- BOSE, R. y VAN DER AALST, W. (2009). Abstractions in process mining: A taxonomy of patterns. *Business*

Process Management, (pp. 159–175).

- BRATOSIN, C. (2011). *Grid architecture for distributed process mining.* PhD thesis., Technische Universiteit Eindhoven, Eindhoven, Eindhoven, The Netherlands.

- BURATTIN, A. y SPERDUTI, A. (2011). PLG: A framework for the generation of business process models and their execution logs. In *Business Process Management Workshops* (pp. 214–219).: Springer. 00028.

- DE WEERDT, J., DE BACKER, M., VANTHIENEN, J., y BAESENS, B. (2012). A multi- dimensional quality assessment of state-of-the- art process discovery algorithms using real- life event logs. *Information Systems*, 37, 654–676.

- GUNTHER, C. (2009). *Process Mining in Flexible Environments.* PhD thesis, Eindhoven University of Technology, Eindhoven.

- GUNTHER, C. y VAN DER AALST, W. (2007). Fuzzy mining - adaptive process simplification based on multi-perspective metrics. In *Business Process Management*, volume 4714 LNCS of *5th International Conference on Business Process Management, BPM 2007* (pp. 328–343). Brisbane.

- MOLKA, T., GILANI, W., y ZENG, X.-J. (2013). Dotted chart and control-flow analysis for a loan application process. In M. L. Rosa y P. Soffer (Eds.), *Business Process Management Workshops*, number 132 in Lecture Notes in Business Information Processing (pp. 223–224). Springer Berlin Heidelberg.

- OUTMAZGIN, N. y SOFFER, P. (2014). A process mining-based analysis of business process work-arounds. *Software & Systems Modeling*, (pp. 1–15). 00000.

- RUSSELL, S. J., NORVIG, P., CANNY, J. F., MALIK, J. M., y EDWARDS, D. D. (1995). *Artificial intelligence: a modern approach*, volume 2 of *Prentice Hall Series in Artificial Intelligence.* Prentice hall Englewood Cliffs, 2 edition.

- VAN DER AALST, W. (2011). *Process Mining. Discovery, Conformance and Enhancement of Business Processes.* Springer, Heidelberg, Dordrecht, London et. al.

- VAN DER AALST, W., WEIJTERS, A. J. M. M., y MARUSTER, L. (2004). Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9), 1128–1142.

- VERBEEK, H., BUIJS, J., VAN DONGEN, B., y VAN DER AALST, W. (2011). *XES, XESame, and ProM 6*, volume 72 LNBIP of *CAiSE Forum 2010 on Information Systems Evolution.* Hammamet.

- WESKE, M. (2007). *Business Process Management. Concepts, Languages, Architectures*, volume 368. Leipzig, Alemania: Springer-Verlag Berlin Heidelberg.

- YZQUIERDO-HERRERA, R. (2012). *Modelo para la estimación de información ausente en las trazas usadas en la minería de proceso*. PhD thesis, Universidad de las Ciencias Informáticas.

- YZQUIERDO-HERRERA, R., SILVERIO-CASTRO, R., y LAZO-CORTÉS, M. (2013). Sub-process discovery: Opportunities for process diagnostics. In G. Poels (Ed.), *Enterprise Information Systems of the Future*, number 139 in Lecture Notes in Business Information Processing (pp. 48–57). Springer Berlin Heidelberg.