Detección de Roles en Sistemas de Gestión de Procesos

Vanesa Pinilla, Silvia Schiaffino, Analía Amandi

ISISTAN, Facultad de Ciencias Exactas, UNCPBA, Tandil, Argentina
CONICET, Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina
{vpinilla, sschia, amandia}@exa.unicen.edu.ar

Abstract. Los sistemas de gestión de procesos se han hecho populares en muchos dominios ya que permiten la gestión íntegra de los procesos dentro de las organizaciones. A pesar de las ventajas que su utilización representa, se han encontrado muchos problemas. Para ayudar a aminorar estos problemas ha surgido lo que se conoce como Minería de Procesos. Su objetivo es extraer información de los logs de ejecución con el fin de generar conocimiento adicional que ayude a mejorar el diseño y la ejecución de los procesos. En este trabajo se presenta un enfoque de la Minería de Procesos aplicado a la detección de roles organizacionales utilizando el algoritmo EM para *clustering*, el cual resultó propicio para este fin luego de compararlo con otras técnicas.

Keywords: Minería de Procesos, *Workflows*, *Clustering*.

1 Introducción

Existe actualmente una gran cantidad de sistemas de información que permiten facilitar el manejo de los procesos llevados a cabo dentro de una organización a través de Sistemas de Gestión de Procesos o *Workflows*, tales como los ERP (Enterprise Resource Planning), BPM (Business Process Management) y WfMS (Workflows Management Systems). Estos sistemas proveen herramientas que facilitan la administración de dichos procesos, de las personas involucradas en cada instancia y de las tareas que participan. Idealmente estos sistemas se encargan de proveer todas las etapas del ciclo de vida de la administración de procesos: (re)diseño – configuración – ejecución – control – diagnóstico. Sin embargo el enfoque no es meramente comercial, pues existe una gran cantidad de sistemas que yacen sobre diferentes dominios como es el caso de los sistemas que se aplican en el contexto medicinal, en sistemas biológicos y en ambientes educativos. Todos ellos tienen la particularidad de ser sistemas orientados a procesos y por ello registran eventos mientras un proceso es ejecutado. Estos registros son almacenados en archivos denominados logs de ejecución, lo que permite la persistencia de este historial de ejecuciones. Cada entrada del log contiene diferente información de acuerdo al sistema que lo genera, pero frecuentemente posee información acerca del comienzo y fin de una actividad y del recurso que lleva a cabo dicha actividad.

La mayoría de los sistemas de gestión de procesos creados en el pasado han sido diseñados para procesos estáticos y estructurados para ser ejecutados en un medio

estable. Sin embargo, hoy en día los sistemas de gestión de procesos pueden ser utilizados en escenarios altamente dinámicos. A su vez los procesos fueron creciendo en complejidad y su ciclo de vida se fue acortando. Por dichas razones es que el diseño de los procesos resulta una tarea complicada y demanda un excesivo tiempo, además de que siempre se encuentran discrepancias entre el proceso diseñado y el proceso real.

Para solucionar estos inconvenientes surge la Minería de Procesos (*Process Mining*), conocida también como Minería de Flujos de Trabajo (*Workflow Mining*). Este tipo de minería provee métodos y técnicas para obtener conocimiento sobre procesos estructurados. La mayoría de las investigaciones en esta área se centran en descubrir flujos de control, es decir en construir modelos de procesos basados en el log, pero otros aspectos han sido descuidados, como por ejemplo los aspectos organizacionales y la interacción entre trabajadores. Esta área se conoce como Minería Organizacional sobre Procesos.

Debido a que el comportamiento humano es fundamental en las organizaciones, pues incide directamente en la *performance* de un proceso, es que la Minería Organizacional sobre Procesos resulta atractiva en este sentido. Bajo este enfoque se intenta descubrir conocimiento organizacional en cualquier ambiente donde las personas juegan un papel fundamental, es decir donde los procesos no son completamente controlados por sistemas. Este conocimiento resulta interesante ya que permite a los administradores e ingenieros de procesos entender las estructuras organizacionales y redes sociales, logrando potenciales mejoras a los procesos involucrados. Un aspecto relevante en los ambientes organizacionales es el rol que tienen asociado los recursos participantes en un proceso. En este contexto entendemos por rol a una posición laboral con la suficiente autoridad y responsabilidad para llevar a cabo ciertas funciones dentro de una organización, en nuestro caso particular para ejecutar una cierta tarea.

Este trabajo se sitúa dentro del área de Minería Organizacional sobre Procesos, implementando una técnica de Inteligencia Artificial para la detección de roles, valiéndose para ello de la similitud entre tareas. En este contexto un rol estaría conformado por un grupo de recursos que ejecutan tareas similares, por lo tanto el desafío de descubrir roles puede ser alcanzado aplicando técnicas de *clustering*. Para ello, primero se realizaron pruebas con tres diferentes técnicas de *clustering*, y de los resultados obtenidos se seleccionó el algoritmo Expectation Maximization (EM) [3]. En el enfoque aquí propuesto se implementó dicho algoritmo y luego para optimizar la obtención de roles se reagruparon aquellos clusters conformados por los mismos recursos. Luego se realiza un análisis de la relación existente entre los roles previamente detectados y del grado de participación de éstos con cada tarea ejecutada.

El resto del trabajo se organiza de la siguiente manera. En la sección 2 se presentan algunos trabajos relacionados. En la sección 3 se presenta una visión general sobre la Minería de Procesos y la Minería Organizacional sobre Procesos y luego se describe nuestra propuesta para la detección de roles en procesos estructurados. La sección 4 describe un proceso tomado como ejemplo para realizar las pruebas. La sección 5 muestra el resultado de una comparación realizada entre algunas técnicas de *clustering* aplicadas sobre el caso de estudio detallado en la sección anterior. En la sección 6 se presenta un análisis de los resultados experimentales obtenidos con el enfoque aquí propuesto. Finalmente la sección 7 presenta nuestras conclusiones.

2 Trabajos Relacionados

La detección de roles relacionada al área de Minería de Procesos ha sido objeto de estudio de algunos autores. En [11] se presenta un método para la obtención de roles donde se intenta agrupar subconjuntos de participantes ejecutando las mismas tareas, pero solo se introduce una noción del enfoque propuesto. Luego en [10] se presenta una versión más detallada del método propuesto anteriormente, donde se utiliza un algoritmo de *clustering* aglomerativo, pero a diferencia de nuestra propuesta parte de tomar como un vector de valores 0 y 1 al conjunto de actividades que un recurso lleva a cabo en un proceso, es decir que inicialmente parte de un vector de actividades para cada recurso que participa en un proceso. En nuestro trabajo en cambio, se forma un vector de dos posiciones, compuesto por la actividad y por el recurso que originó dicha actividad, para cada evento producido en una instancia de proceso.

En [13][14] se aplica Minería de Procesos pero a diferencia de nuestra propuesta busca obtener redes sociales para tal fin. Allí los autores plantean un enfoque para el descubrimiento de roles, valiendose de la construccion de un perfil para cada recurso en base a cuán frecuente estos recursos ejecutan ciertas actividades. Luego es posible calcular ciertas métricas entre pares de perfiles para medir la distancia entre ambos. Tales métricas son utilizadas para la construcción de un sociograma que puede ser analizado mediante el uso de técnicas existentes provenientes del Análisis de Redes Sociales.

3 Visión General de Nuestra Propuesta

Nuestro trabajo se centra en la detección de estructuras organizacionales, teniendo en cuenta dos objetivos principales de la Minería Organizacional: la detección de estas estructuras y la comunicación entre los recursos. Nuestra propuesta garantiza la detección de estructuras organizacionales, específicamente de roles, y también plantea una visión sobre cuán relacionados están dichos roles y especifica el grado de participación de cada rol para cada tarea que ejecuta.

A continuación se introducirán los temas de Minería de Procesos y Minería Organizacional sobre Procesos, luego se presentarán algunos conceptos para la detección de roles y finalmente se detallará el enfoque utilizado en este trabajo.

3.1 Minería de Procesos y Aspectos Organizacionales

Los procesos generalmente son modelados mediante *workflows* (flujos de trabajo) y soportados por Sistemas de Gestión de Procesos. Un *workflow* puede ser definido como la automatización de un procedimiento, donde información, documentos y/o tareas son pasados de un participante a otro siguiendo ciertas de reglas [9]. También puede ser visto como una representación automatizada de un proceso de negocio [2]. En un *workflow* es necesario definir cómo se estructuran las tareas, cuál es su orden, cómo se sincronizan, cómo fluye la información que éstas soportan y cómo se hace el seguimiento de las mismas.

Las tareas que conforman un proceso, son registradas junto con algunos otros datos relevantes de las instancias de proceso. Estos datos se almacenan de alguna manera y se conocen como "logs de ejecución", los cuales son utilizados para satisfacer el objetivo de la Minería de Procesos que puede resumirse como: "extraer información de ejecuciones anteriores con el fin de obtener conocimiento adicional que ayude a tomar decisiones en futuras instancias del proceso" [15][16][4].

Dentro del área de Minería de Procesos estudiaremos la perspectiva social, comúnmente conocida como Minería Organizacional sobre Procesos [12][5]. La Minería Organizacional sobre Procesos es aplicable a diversas situaciones, especialmente cuando los procesos no son completamente controlados por sistemas, donde las personas juegan un rol fundamental. El descubrimiento de conocimiento en esta área, como son las estructuras organizacionales y redes sociales, permite a los administradores de procesos entender y mejorar los procesos. Un ejemplo de ello es la aplicación de redes sociales en el contexto de procesos, las cuales muestran las estructuras de comunicación en una organización.

3.2 Detección de Roles

El Aprendizaje Automático o Aprendizaje de Máquina es una rama de la Inteligencia Artificial (IA) que tiene por objetivo el desarrollo de técnicas que permitan a las computadoras aprender a partir de información no estructurada suministrada en forma de ejemplos. El aprendizaje no supervisado es una de las formas del aprendizaje automático, pero su particularidad distintiva es que no se dispone de antemano de información sobre las categorías de esos ejemplos – a diferencia del aprendizaje supervisado. Un ejemplo común de aprendizaje no supervisado es la aplicación de técnicas de *clustering* [7][8]. En *clustering* se cuenta con un conjunto de objetos de entrada pero se desconoce tanto el número de clases en que es razonable particionar dicho conjunto como también a qué clase pertenece cada objeto.

El problema de descubrimiento de roles dentro de logs de ejecución de procesos puede ser visto como un problema de aprendizaje automático. Debido a que generalmente no se cuenta con información a priori acerca de los roles asociados a cada recurso, se tratará como un problema dentro del aprendizaje no supervisado aplicando técnicas de *clustering*. De esta manera, el objetivo que se persigue es agrupar en un mismo *cluster* todos aquellos recursos cuya similitud está dada por las tareas que realizan, luego dichos *clusters* representarán los roles que se pretende obtener.

3.3 Uso de EM para Detección de Roles

El algoritmo Expectation Maximization (EM) forma parte de un conjunto de modelos conocidos como Modelos de Mezcla Finita, los cuales se pueden utilizar para segmentar conjuntos de datos [3]. En estadística, EM es un método para encontrar la probabilidad máxima dentro de modelos estadísticos, donde el modelo tiene valores de variables no conocidas o latentes. EM es un método iterativo que alterna entre ejecutar un paso de iteración (E) y un paso de maximización (M). En el paso E se

estima la distribución de probabilidad de los datos no conocidos, en base los valores conocidos de las variables observadas y a la estimación de los parámetros actuales. En el paso M se re estiman los parámetros encontrados con el objetivo de maximizar los valores obtenidos en el paso anterior. Luego estos parámetros son utilizados como entrada para ejecutar nuevamente el paso E. Este proceso se repite hasta que el algoritmo EM converge a un mínimo.

El algoritmo EM aplicado en técnicas de *clustering* se usa para computar la probabilidad de que cada dato es miembro de alguna clase, donde el paso M se refiere a maximizar dichas probabilidades. El objetivo de EM aplicado a *clustering* es estimar la media y la desviación estándar para cada *cluster* y así maximizar la función de bondad de los datos observados [18]. EM puede verse como un método de *clustering* probabilístico donde se trata de obtener la FDP (Función de Densidad de Probabilidad) desconocida a la que pertenecen el conjunto completo de datos. Esta FDP se puede aproximar mediante una combinación lineal de N componentes, definidas a falta de una serie de parámetros $\{\theta\} = U \{ \theta_j V_j = 1 \ldots N\}$, que son los que hay que averiguar. FDP puede ser formulada como sigue:

$$P(x) = \sum_{j=1}^{N} \pi_j p(x;\theta_j) \text{ donde } \sum_{j=1}^{N} \pi_j = 1 \quad (1)$$

donde π_j son las probabilidades a priori de cada *cluster* cuya suma debe ser 1, P(x) denota la FDP y $p(x; \theta_j)$ la función de densidad del componente j. Cada *cluster* se corresponde con las respectivas muestras de datos que pertenecen a cada una de las densidades que se mezclan.

El ajuste de los parámetros del modelo requiere alguna medida de su bondad o *likelihood*, es decir de cuán bien encajan los datos sobre la distribución que los representa. Se trata entonces de estimar los parámetros buscados θ, maximizando el *likelihood.* Normalmente, se calcula el logaritmo de esta medida *likelihood*, conocido como *log-likelihood* ya que analíticamente es más fácil de calcular. La solución obtenida es la misma, gracias a la propiedad de monotonicidad del logaritmo. La forma de esta función log-likelihood es:

$$L(\theta,\pi) = \log \prod_{n=1}^{NI} P(x_n) \quad (2)$$

donde NI es el número de instancias, que se suponen independientes entre si.

Luego se ejecutan los pasos de E y M respectivamente. Después de una serie de iteraciones, el algoritmo EM tiende a un máximo local de la función L. Finalmente se obtendrá un conjunto de *clusters* que agrupan el conjunto original. Cada uno de estos *clusters* estará definido por los parámetros de una distribución normal.

3.4 Implementación de Nuestra Propuesta

En este trabajo la estrategia de *clustering* fue implementada como un plug-in de la herramienta ProM[1] [17] utilizando la API de Weka[2]. La herramienta ProM es un *framework* desarrollado por el grupo de Minería de Procesos de la Universidad Técnica de Eindhoven, el cual posee varios plug-ins centrados tanto en la minería de procesos como en el análisis de logs. ProM es una aplicación de código abierto desarrollada en JAVA que soporta perfectamente el enfoque aquí propuesto. A continuación, se describen los pasos ejecutados para la detección de roles:

1. Filtrar el log, dejando solo los atributos relacionados a los originadores y las actividades que éstos realizan, para cada evento registrado. Luego representar estos pares de atributos como un vector `[recurso, tarea]`, para todas las instancias del proceso.
2. Ejecutar el algoritmo EM, entrenándolo con todas las instancias de log. La implementación del algoritmo EM utilizada decide la cantidad de *clusters* a generar basándose en la estrategia *cross validation*.
3. En base a los resultados de la ejecución de EM, crear una estructura organizacional con los roles obtenidos, los recursos pertenecientes a cada rol y las tareas asociadas a ellos.
4. Para expresar cuán abocado se encuentra un rol a cada tarea que ejecuta, calculamos la frecuencia con que se realizan (cantidad de veces que cada rol realiza la tarea sobre el total de veces que el rol ejecutó cualquier tarea). Esta métrica se expresa en la estructura organizacional mencionada en el paso anterior.
5. Una vez obtenidos los roles, se genera un sociograma para expresar la relación existente entre ellos. Para ello, asumimos que dos roles se encuentran relacionados, si los originadores de éstos realizan al menos una actividad en común.

Para medir cuantitativamente este nexo nos valemos de una expresión vectorial, donde cada elemento del vector representa la frecuencia con que un rol realiza cada una de las tareas. La dimensión de estos vectores siempre será la misma, igual al número total de tareas, independientemente del rol que representen. Si un rol no realiza una tarea, el correspondiente valor en esa posición será nulo. Con esta representación podemos utilizar una métrica para cuantificar el grado de relación entre roles, en nuestro caso se ha utilizado la distancia Euclídeana.

4 Caso de Estudio

Se seleccionó un ejemplo de un proceso de reparación de teléfonos, provisto por la herramienta ProM. El mismo ejemplo será utilizado a lo largo de este trabajo para las pruebas realizadas con otros algoritmos y para mostrar los resultados obtenidos con nuestra propuesta. El archivo del ejemplo es "repairExample.mxml" [3]. Las pruebas fueron corridas también con otro conjunto de datos, mostrando aquí solo un ejemplo por razones de espacio.

[1] http://prom.win.tue.nl/tools/prom/
[2] http://www.cs.waikato.ac.nz/_ml/weka/
[3] http://prom.win.tue.nl/research/wiki/_media/tutorial/repairexample.zip

En la Fig. 1 se muestra una representación gráfica del proceso seleccionado, el cual se describe a continuación:

1. El proceso comienza registrando el teléfono enviado por un usuario.
2. Luego es enviado al departamento de "Detección de errores". Aquí el teléfono es analizado y el defecto es categorizado.
3. Identificado el problema, el teléfono es enviado al departamento de "Reparación" y se envía una carta al cliente. Reparación tiene dos equipos, uno repara defectos simples y otro repara defectos complejos. Sin embargo algunos defectos pueden ser reparados por ambos equipos.
4. Una vez reparado el teléfono, es enviado al departamento de "Calidad". Allí el dispositivo es analizado para ver si el problema se arregló o no. Si no es así, es enviado nuevamente al departamento de reparación (paso 2). Si el problema se arregló, el caso es archivado y el dispositivo se envía al cliente.

El dataset utilizado tiene 1104 instancias de procesos, 11855 eventos (un evento por cada tarea instanciada y sus atributos relacionados) y 13 diferentes recursos que ejecutan tareas (SolverS1, SolverS2, SolverS3, SolverC1, SolverC2, SolverC3, Tester1, Tester2, Tester3, Tester4, Tester5, Tester6, System). Los roles no están documentados en el log, pero el dataset seleccionado es un ejemplo diseñado especialmente para realizar pruebas, por lo cual se puede deducir intuitivamente los roles que deberíamos detectar. Estos roles son: recursos que testean, recursos que resuelven un tipo de problema, recursos que resuelven otro tipo de problema y encargado del proceso.

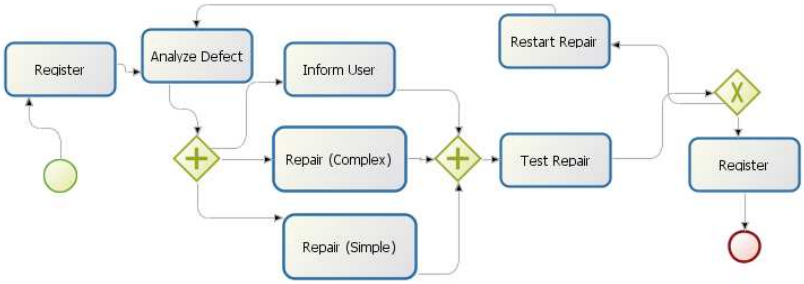

Fig. 1. Diagrama del proceso de reparación de teléfonos

5 Evaluación Empírica de Técnicas de *Clustering*

Como un estudio previo a la implementación de nuestra propuesta, se realizó una evaluación empírica acerca de un grupo de algoritmos de *clustering* conocidos, utilizando el software Weka. El objetivo de este estudio fue analizar cuál de estos algoritmos detectaba mejor los roles buscados. Los algoritmos evaluados fueron Cobweb [6], CLOPE [19] y EM [3]. Estos algoritmos fueron seleccionados debido a que no requieren conocer de antemano el número de *clusters* a obtener, ya que en nuestro caso no es posible predecir a priori la cantidad de roles buscados. A continuación se describirán los resultados de la evaluación realizada para cada uno de estos algoritmos.

5.1 Desempeño del Algoritmo Cobweb

Este algoritmo es de tipo jerárquico y se caracteriza por utilizar aprendizaje incremental. Al ejecutarse forma un árbol donde las hojas representan los segmentos y el nodo raíz engloba el conjunto completo de datos de entrada. Además COBWEB pertenece a los métodos de aprendizaje conceptual o basado en modelos (cada *cluster* se considera un modelo que puede describirse intrínsecamente).

En la Fig. 2 puede observarse los resultados obtenidos utilizando los parámetros por defecto. Este agrupamiento escapa el objetivo al que apuntamos; proveyendo más roles de los necesarios a simple vista. Un ejemplo básico de ello es agrupar {SolverC1, SolverC2, SolverC3} ya que comparten la realización de la tarea "Repair (Simple)"; en el caso de este algoritmo los coloca a todos ellos en *clusters* separados.

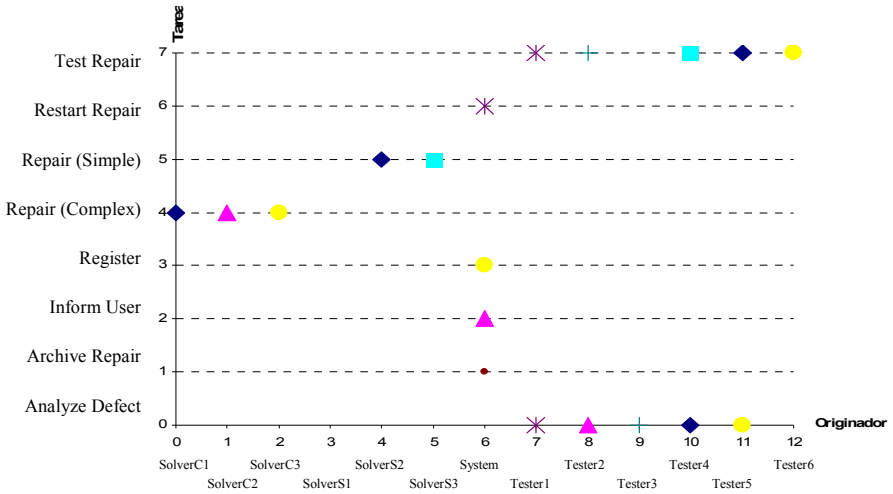

Fig. 2. *Clustering* efectuado con el algoritmo CobWeb

5.2 Desempeño del Algoritmo CLOPE

Este algoritmo se propuso basado en la idea intuitiva de incrementar el radio ancho-a-alto del histograma del *cluster*, intentando así incrementar la coincidencia intra-*cluster*. Esta idea se lleva a cabo mediante un parámetro de repulsión que controla la limitación de las instancias en un *cluster*. Al utilizar distintos valores para este parámetro se obtuvieron distintos resultados, por ejemplo al utilizar un valor de repulsión de 2.6 nos agrupó los "Testers" en *clusters* separados, lo cual nos indica que las agrupaciones no son las buscadas. Al modificar este valor de repulsión por 0.8 se obtuvieron 4 *clusters*, la cantidad de grupos que buscamos, pero genera una distribución que no es la esperada, por ejemplo a "Tester 5" lo coloca en un *cluster* distinto a los demás "Testers". Los resultados obtenidos para este valor de repulsión pueden verse en la Fig. 3.

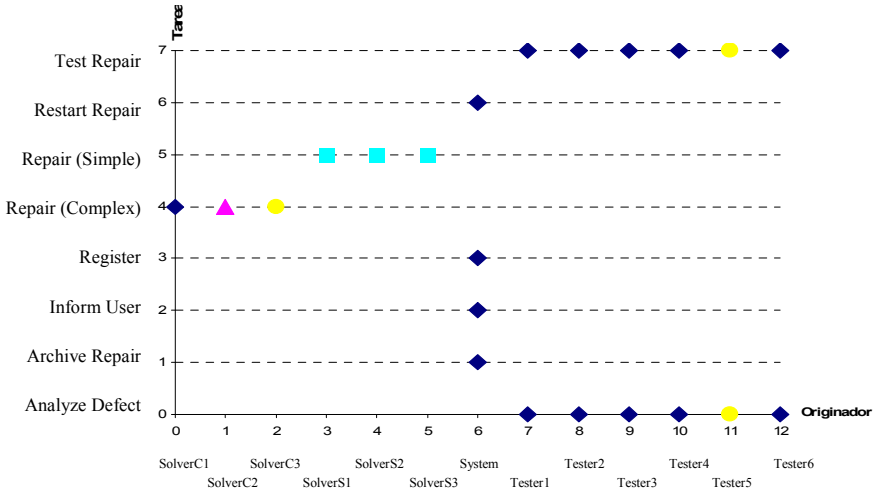

Fig. 3. *Clustering* efectuado con el algoritmo CLOPE (repulsión de 0.8)

5.3 Desempeño del Algoritmo EM

Ejecutando el algoritmo EM con los parámetros por defecto se obtuvieron los resultados mostrados en la Fig. 4. Allí puede observarse que la distribución de los datos es la esperada. En este caso, al tener cada tipo de originador tareas específicas asignadas, por ejemplo todos los "Testers" se dedican a "Analyze Defect" y a "Test Repair", los agrupa a todos ellos en un único rol.

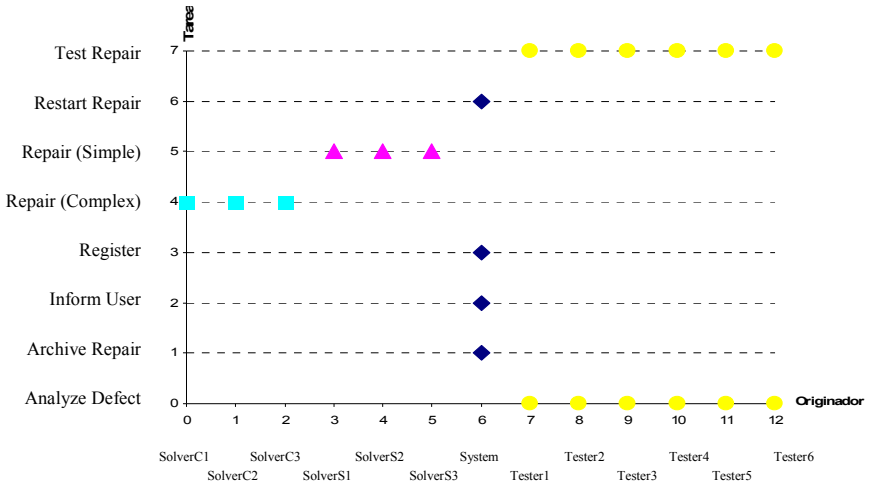

Fig. 4. *Clustering* efectuado con el algoritmo EM

5.4 Discusión

Dado el resultado de los experimentos antes mostrados, podemos decir que el algoritmo que mejor se adapta a nuestras exigencias es EM, pues es el que mejor

detecta los roles buscados. Es preciso aclarar que el dataset del ejemplo tiene una distribución homogénea de los datos, es decir que se eligió un log en el cual los originadores realizan de manera equitativa sus tareas, pues a simple vista es más fácil detectar el error en la formación de *clusters* o roles. Esto también se ve facilitado por el hecho de que no existen tareas compartidas entre los supuestos *clusters*, en cuyo caso sería más difícil detectar roles de manera intuitiva. Además se probó con un dataset heterogéneo creado especialmente para tal fin, donde los roles pueden compartir la realización de algunas tareas. Los resultados obtenidos en este caso también fueron mejor con el algoritmo EM, pero no se documentan en este trabajo por razones de espacio.

EM nos proporciona *clusters* con una alta similitud intra-cluster y una baja similitud inter-cluster. Esto, sumado al hecho de que desconocemos la cantidad de roles deseados, satisface nuestras necesidades. Un punto a mencionar es que, de los tres algoritmos analizados, EM resultó ser el más costoso en cuanto a tiempo de ejecución, pero esto se ve compensado con los buenos resultados obtenidos. El tiempo de ejecución de algoritmo CoWeb fue de 60 seg., el de CLOPE fue de 53 seg. y el de EM fue de 42 min. y 10 seg.

6 Análisis de Resultados Experimentales Obtenidos con Nuestra Propuesta

Para evaluar el desempeño de nuestra propuesta se analizó el log de un proceso de reparación de teléfonos, el cual fue detallado en la Sección 3. Además se desarrolló un plug-in para la herramienta ProM [17] utilizando la API de Weka.

Luego de hacer correr el plug-in para el log utilizado como caso de estudio, se obtuvieron los resultados mostrados en la Fig. 5, donde los rectángulos azules representan los roles que fueron detectados. Aquí se sugiere la formación de 4 roles, y se muestra el grado de dedicación que empleará cada rol sobre las tareas que deben realizar, lo cual puede verse indicado en los valores de cada arco. Por ejemplo, podemos observar que para el rol "Role 1" el 57,7 % de las veces ejecutó la tarea "Test Repair" y el 42,3% de las veces ejecutó la tarea "Analyze Defect", es decir que los recursos que pertenecen a dicho rol están autorizados a ejecutar ambas tareas.

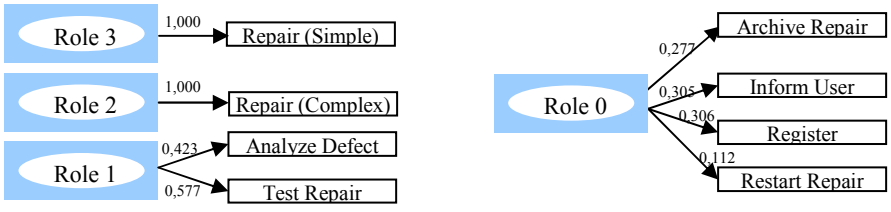

Fig. 5. Roles detectados con la aplicación del algoritmo de *clustering* EM

Otro aspecto importante a analizar es la relación entre las entidades descubiertas. Esto es expresado en la representación grafica de la Fig. 6, donde puede observarse que no existen relaciones entre los roles detectados para el ejemplo citado. Esto se

debe a que los roles no ejecutan tareas en común. Detectar estas relaciones podría resultar beneficioso para un analista o gerente, por ejemplo para redefinir los permisos de acceso a las distintas tareas o realizar un análisis de la incidencia que tiene un rol con otro, pudiendo ver por ejemplo cómo repercutiría la desaparición de un rol sobre los demás.

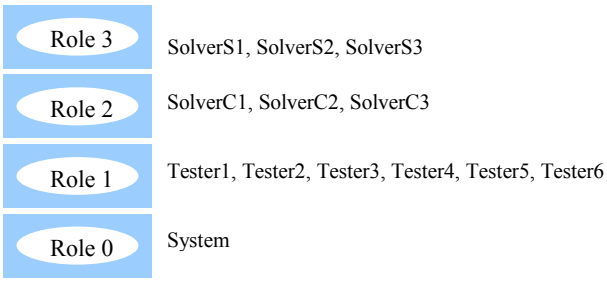

Fig. 6. Relación entre los roles detectados

7 Conclusiones

En este trabajo hemos presentado un enfoque para la detección de roles en logs generados por sistemas de gestión de proceso. La técnica de *clustering* utilizada para tal fin fue comparada con los resultados obtenidos con otras técnicas de *clustering*, resultando el algoritmo EM para *clustering* el más apropiado y en el cual se obtuvieron los mejores resultados.

El enfoque desarrollado en este trabajo se vale del algoritmo EM para luego reagrupar los resultados obtenidos, además se calculan métricas de frecuencia para analizar las relaciones entre los roles obtenidos y el grado de participación de cada rol en las tareas que realiza. No tenemos conocimiento de que el algoritmo EM haya sido utilizado para la obtención de roles en el área de Minería Organizacional sobre Procesos. Algunas ventajas de nuestro trabajo son, que al ser desarrollado como un plug-in para ProM permite la integración con varias técnicas de análisis y minería de procesos, pudiendo un mismo log ser analizado desde diferentes perspectivas no solo desde la perspectiva organizacional. Además permite ser extendido agregando nuevas técnicas.

Como trabajo futuro, planeamos comparar el enfoque aquí propuesto con las técnicas de detección de roles utilizadas por otros autores, mencionadas en la sección 2, para hacer un análisis de las diferencias obtenidas.

Referencias

1. A.K. Alves de Medeiros and A.J.M.M. Weijters, “Workflow Mining: Current Status and Future Directions,” On The Move to Meaningful Internet Systems 2003: CoopIS,DOA, and ODBASE, volume 2888 of Lecture Notes in ComputerScience, Springer-Verlag, 2003, pp. 389–406.

2. K. Belhajjame, G. Vargas-Solar, and C. Collet, "A flexible workflow model for process-oriented applications," in the 2nd international conference on web information systems engineering, WISE'2001, Kyoto-Japan, 2001, pp. 3-6.
3. A.P. Dempster, M.N. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 39, 1977, pp. 1–22.
4. C.A. Ellis, K. Kim, and A.J. Rembert, "Workflow Mining: Definitions, Techniques, and Future Directions," L. Fischer, Ed., Future Strategies Inc., 2006, pp. 213-228.
5. A. Gao, Y. Yang, M. Zeng, J. Zhang, and Y. Wang, "Organizational Structure Mining Based on Workflow Logs," 2009 International Conference on Business Intelligence and Financial Engineering, Beijing, China: 2009, pp. 455-459.
6. D.H. Fisher, "Knowledge acquisition via incremental conceptual clustering," Machine Learning, vol. 2, 1987, pp. 139-172.
7. A.K. Jain, M.N. Murty, and P.J. Flynn, "Data clustering: a review," ACM Computing Surveys, vol. 31, 1999, pp. 264-323.
8. S.J. Hanson and M. Bauer, "Conceptual clustering, categorization, and polymorphy," Machine Learning, vol. 3, 1989, pp. 343-372.
9. D. Hollingsworth, Workflow Management Coalition Specification: The Workflow Reference Model, Hampshire, UK: Workflow Management Coalition, 1995.
10. A.J. Rembert and C.(. Ellis, "An initial approach to mining multiple perspectives of a business process," The Fifth Richard Tapia Celebration of Diversity in Computing Conference: Intellect, Initiatives, Insight, and Innovations, Portland, Oregon: ACM, 2009, pp. 35-40.
11. A.J. Rembert, "Comprehensive workflow mining," ACM-SE 44: Proceedings of the 44th annual Southeast regionalconference, New York, NY, USA: ACM, 2006, pp. 222–227.
12. M. Song and W.M.P. van der Aalst, "Towards comprehensive support for organizational mining," Decis. Support Syst., vol. 46, 2008, pp. 300–317.
13. W.M.P. van der Aalst, H.A. Reijers, and M. Song, "Discovering Social Networks from Event Logs," Computer Supported Cooperative Work, vol. 14, 2005, pp. 549–593.
14. W.M.P. van der Aalst and M. Song, "Mining Social Networks: Uncovering Interaction Patternsin Business Processes," Business Process Management, 2004, pp. 244-260.
15. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster, Workflow Mining: Discovering Process Models from EventLogs. IEEE Transactions on Knowledge and Data Engineering, 2003, 16(9):1128-1142.
16. W.M.P. van der Aalst, B.F. van Dongen, J.A. Herbst, G. Schimm, and A.J.M.M. Weijters, "Workflow mining: a survey of issues and approaches," Data Knowl. Eng., vol. 47, Nov. 2003, pp. 237–267.
17. W.M.P. van der Aalst, B.F.V. Dongen, C. Günther, R.S. Mans, A.K.A.D.M.A. Rozinat, V. Rubin, M. Song, H.M.W. (Eric)Verbeek, and A.J.M.M. Weijters, "ProM 4.0: Comprehensive Support for Real Process Analysis," ICATPN, J. Kleijn and A. Yakovlev, Eds., Springer, 2007, pp. 484-494.
18. I. Witten, Data mining: practical machine learning tools and techniques with Java implementations, San Francisco Calif.: Morgan Kaufmann, 1999.
19. Y. Yang, X. Guan, and J. You, "CLOPE: a fast and effective clustering algorithm for transactional data," Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '02, Edmonton, Alberta, Canada: 2002, p. 682.