ISSN 2395-8618

TrazasBP: A Framework for Business Process Models Discovery Based on Execution Cases

Hugo Ordoñez, Armando Ordóñez, Victor Buchelli, and Carlos Cobos

Abstract—**Execution of business processes generates data, which are commonly recorded in logs. Historical information of execution cases may be used for recommending future execution paths. This is useful when the control flow of the process is not known by the user. We present TrazasBP, a framework for BP indexing and searching based on execution cases. It indexes BPs based on execution cases (traces) retrieved from log files. TrazasBP not only takes into account the textual information of BP elements, but also the causal dependence between these elements. Furthermore, due to its low computational cost, TrazasBP may be used as indexing mechanism in order to reduce the search space. Experimental evaluation shows promising values of graded precision, recall and F-measure when compared with results obtained from human search.**

Index Terms—**Business process, execution cases, Logs, repository, evaluation**

I. Introduction

INNOVATON in products and services is mandatory for competitiveness in today's market. Commonly, tasks and functions related to commercial activities of companies are represented within Business processes (BP) [1]. A BP consists of a set of logically-related tasks executed sequentially in order to generate valid outputs for the business. BP executions must follow guidelines given by internal policies, standards, best practices and laws. For example, doctors should only perform surgeries within the scope of their specialty area. Furthermore, this surgery should be preceded by an authorization from the patient and the hospital. Another example, in sales processes, an order should be archived only after customer confirms reception of ordered items [2].

Manuscript received on October 25, 2017, accepted on February 27, 2018, published on June 30, 2018.

H. Ordonez is with the Research Laboratory in Development of Software Engineering of the Universidad San Buenaventura, Colombia (e-mail: haordonez@usbcali.edu.co).

A. Ordonez is with the Foundation University of Popayan, Colombia (corresponding author, e-mail: jaordonez@unicauca.edu.co).

V. Buchelli is with the Foundation University of Valle, Colombia (e-mail: victor.bucheli@correounivalle.edu.co).

C. Cobos is with the Foundation University of Cauca, Colombia (e-mail: ccobos@unicauca.edu.co).

Execution of tasks and processes generates a set of data which is recorded in logs [3]. Logs contain information of executed processes, namely: roles, resources, participants, interaction with other systems, transactions performed and execution dates, among other data. When a BP is initiated, an instance (execution) of the BP is created, therefore Logs store information of many instances or executions of the same process [4]. Specifically, historical execution traces containing information of actually executed instances are known as *execution cases*. These execution cases contain information of the path followed by the control flow during actual execution of a BP instance [5].

This paper presents TrazasBP, a framework for BP indexing and searching based on execution cases. TrazasBP indexes BPs based on execution cases (traces) retrieved from log files. A log file is created when a BP is executed for the first time, and it is updated by adding new execution cases as executions are carried out. Execution cases register information about a specific BP execution (i.e. what activities were executed at a certain moment in time during BP execution) [6]. Thus, a BP contains only one log file, but multiple execution cases included in this file. TrazasBP considers in addition to textual information of BP elements, the causal dependence between these elements. Furthermore, due to its low computational cost, TrazasBP may be used as indexing mechanism in order to reduce the search space.

The main contributions of TrazasBP are twofold: i) it may be used for indexing generation based on the execution cases, and ii) TrazasBP allows ranking a set of executed BPs in concordance with their similarity with a query BP. Historical information of *execution cases* may be used for recommending future execution paths. This is useful when the control flow of the process is not known by the user, for example, when the doctor doesn't know about new treatments or when a company cannot foresee the behavior of potential customers [7].

The rest of the paper is organized as follows: Section 2 presents related works, section 3 describes TrazasBP architecture, Section 4 shows the evaluation and results, finally, conclusions and the implications of the results are given in Section 5.

II. RELATED WORKS

Several approaches for measuring similarity of BP are available in the literature. These approaches are based on different BP characteristics such as: linguistics [8], [9], structure [10], [11] and behavior [12], [13]. As the approach presented here considers similarity of execution cases, therefore this section study approaches considering causal dependency between activities, and common sets of execution traces

Bae et al [14] present a dependency graph to compare two BPs taking into account differences between arcs or edges that links activities in both BPs. This approach does not consider gateway types. Weidlich et al [15] define causal behavioral profiles representing dependencies between activity pairs. Similarity is calculated by identifying activity pairs for which there are corresponding pairs of activities. Then corresponding pairs sharing the same relations are analyzed.

Dijkman et al [16] represent precedence relations between activities as loopback links and causal footprints. Causal footprints are in turn represented as vectors of index terms. This approach builds vectors of high dimension, which increase the computational cost of the method. Other existing approaches consider direct precedence of activities represented as Transition Adjacency Relation [17], n-grams [18], and behavioral profiles [15]. In those cases similarity is calculated analyzing correspondence between direct precedence of activities in the trace.

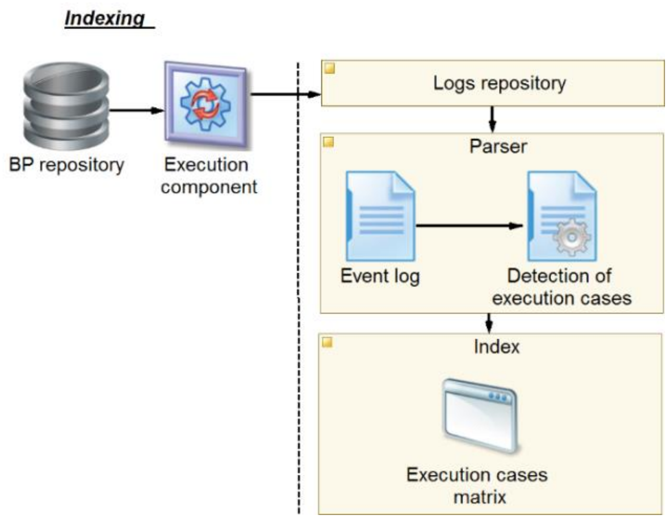


Fig. 1. Components of the indexing phase

Gerke et al [19] , Wang et al [20] compare the compliance between BPs calculating the longest common subsequence of traces, i.e., similarity degree of ordering rules of activities between two BPs. However, this approach is computational expensive when there are large sets of traces. Weerdt et al [6] deal with real execution traces of BPs in order to discover BPs, i.e., this method aims for inferring which BPs can produce such traces. Medeiros et al [9] compare BPs by studying frequency of traces obtained from actual or simulated executions.

TrazasBP integrate characteristics of the aforementioned approaches by considering causal dependence between activities of actual execution cases. Additionally, Due to TrazasBP low computational cost, it may to be used as indexing mechanism preceding other expensive algorithms during BPs similarity calculation; the last is possible because TrazasBP reduces the search space. Next section presents the architecture of TrazasBP and describes its components.

III. ARCHITECTURE OF TRAZAS BP

TrazasBP allows indexing and searching BPs stored in a repository according to their similarity with a query represented as a set of node pairs (PS_q). Three kinds of query options are supported: execution cases, minimal behavior, and log-files (These types of queries are detailed later). TrazasBP works both during indexing phase and querying phase. During indexing phase (See Figure 1), logs are indexed and a matrix *Mec* of execution-cases is generated. Then, during querying phase (See Figure 3), a query (Execution case, minimal behavior, or log-file) is received and processed in order to obtain a set of node pairs. Finally, when the set of node pairs are obtained, the query matrix is generated *Mq* and the repository is ranked. Next both phases are described

3.1 Indexing phase

3.1.1 BP Repository

This repository stores a set of BPs, these BPs are executed in order to generate the execution cases. The current implementation of the repository includes 100 BPs modeled with BPMN (Business Process Modeling Notation). Those BPs were graphically designed by experts of the Telematics Engineering Group of the University of Cauca (Colombia) based on real processes provided by Telco operators in Colombia and examples found in different web sites (e.g the TM Forum2). A real repository of a Telco operator couldn't be used due to privacy and security policies of Telecom operators.

3.1.2 Execution component

This component executes BPs and collects log-files containing execution cases. The current version of this component is implemented using the Bizagi BPM suite which is a popular tool for BP modeling [19]. BPs were executed in the lab (in order to simulate real executions) and log-files were then stored in a second repository named "logs repository".

3.1.3 Log Repository

This repository stores all the log-files obtained from the execution of BPs. Each BP contains only one log-file with multiple execution cases. The current implementation of this repository stores the log-files in the file system.

3.1.4 Parser

This component extracts and processes execution cases from each log-file stored in the "*logs repository*". Here, execution cases are represented as vectors (*execution case vectors*) that associate execution cases with *BPs*. Afterwards, each execution case vector is processed to form pairs of adjacent nodes in order to keep causal relationships. Once node pairs are formed, they are arranged together with node pairs of other execution cases in the same *BP* in order to create a new vector (*node pairs vector*). This procedure is repeated for the entire *BP* repository obtaining one vector of node pairs for each *BP*.

3.1.5 Index

This component processes node pairs vectors and generates an index. First at all, node pairs from each vector are analyzed with the Porter Stemming[10] algorithm that transforms node labels to their lexical root (e.g. words "helping" and "helped" are transformed to their lexical root "help"), later the same algorithm removes special characters, void words, and accents. Next, the indexer creates a "matrix of execution cases" (*Mec*) whose rows are the BPs stored in the repository, and the columns are the node pairs of all the BPs of the repository but avoiding the pairs that are duplicated. The matrix *Mec* is filled by counting the number of times that a pair is found in each BP (i.e. in the vector of node pairs of each BP).

	p_1	p_2	...	p_j	...	p_k
BP_1	0	0	...	2	...	1
BP_2	3	0	...	0	...	0
...	...	...	...	..	...	...
BP_i	2	0	...	3	...	2
...	...	...	...	..	...	...
BP_n	1	5	...	0	...	3

Fig. 2. Example of execution cases matrix

Let $R = \{BP_1, ..., BP_i, ...BP_m\}$ be a repository of BPs. Each $BP_i \in R$ contains a log file $l_i = \{ec_{i1}, ..., ec_{ij}, ...ec_{ik}\}$ that is updated each time the BPi is executed by adding a new execution case ecij . Each execution case is composed of a sequence of BP elements (nodes) which may be activities or gateways (XOR (Join-Split), AND(Join-Split)). These elements are ordered according to the execution flow followed by the BP. The first step in the BP indexing mechanism is to collect all the nodes of each execution case ecij = {n1, ..., np} and form pairs of nodes keeping their causal dependence (i.e., adjacent nodes in the execution case). For example, in ecij the set of node pairs is PSij = {(n1 , n2), (n2 , n3), ..., (ni−1 , ni),(ni , ni+1), ..., (np−1 , np)}.

After collecting pairs of the execution cases of the entire repository, a matrix named "execution cases matrix" Mec is created. In this matrix columns represent node pairs found in the execution cases for the entire repository but avoiding those repeated (i.e. there are not two columns representing identical node pairs), and rows are all the BPs stored in the repository. Therefore, the size of the matrix is m × k, where m is the number of BPs in the repository, and k is the number of all pairs found in execution cases avoiding those which are repeated.

Finally, the matrix *Mec* is completed with the number of times a pair is found in the execution cases of a given *BP*, e.g., if a pair p_j is found three times in the log $l_i \in BP_i$, then the number 3 is inserted on the cell (i; j) (Figure 2). Thus, the index of execution cases is created and represented by the matrix *Mec*. In the present approach, this matrix is similar to the "term-document matrix" of the vector space model in the Information Retrieval (IR) field proposed by Salton in 1989. Therefore, the *Mec* matrix can be normalized in the same way as the "term-document matrix", which is composed of cells w_{ij} representing textual components (in their lexical root) detected in a log-file. Then, each w_{ij} is weighted with the equation 1, where F_{ij} is the observed frequency in the component *j* of the BP_i; $Max(F_i)$ is the highest observed frequency of the BP_i; *N* is the number of *BP* in the repository; and n_j is the number of *BP* in which the execution case *j* has been detected

$$w_{i,j} = \frac{F_{i,j}}{\max(F_i)} \times \log\left(\frac{N}{n_j + 1}\right) \tag{1}$$

3.2 Query phase

First This phase has two functions: firstly, it transforms queries into node pairs in order to create a query matrix (*Mq*). *Mq* contains information about the frequency of each pair in the query. Secondly, this phase ranks BPs according to their similarity to the query. (See Figure 3)

3.2.1 Query Processor

This module receives a query and transforms it into a set of node pairs. The current implementation of the query module supports the following 3 kinds of queries:

Execution case: the query is a textual string that represents a BPs execution case. Therefore, the string must contain a sequence of nodes (activities and gateways) that will be transformed into a set of node pairs.

Minimal behavior of execution flow: the query is a list of node pairs obtained from the execution cases of the BPs in the

ISSN 2395-8618

repository. Then a user can choose a combination of node pairs to build a query.

Log file: the query is a log-file that is processed to identify the execution cases and subsequently the sets of node pairs. In this option, the user can choose one of the found sets of execution cases in order to rank the BPs in the repository that have executed similar execution cases. Once, the query is transformed, the set of node pairs are processed with the "PorterSteeming" algorithm as explained before. Then, the duplicated pairs are counted and inserted in a query vector which contains the number of occurrences of each pair.

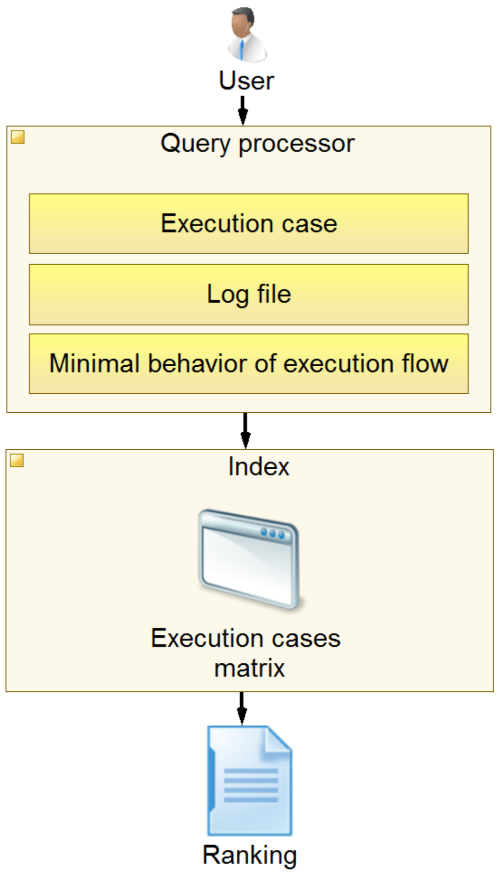


Fig. 3. Components of the indexing phase

3.2.2 Ranking

In this phase the query vector vq and the execution cases matrix Mec are integrated in the query matrix (Mq) as described in section 3.23. The Mq matrix is useful for measuring similarity between each BP of the repository and the query and to produce a ranking of BPs according to this degree of similarity.

3.2.2 Querying the index of execution cases

To query the index of execution cases a query set of node pairs (*PS_q*) is required. The set *PS_q* is processed in order to find repeated node pairs, and to create a query vector *vq* that registers the number of occurrences of each pair. For example, let $PS_q = \{p_{q1}, p_{q2}, \dots p_{qi}, \dots p_{qt}\}$, if $p_{q1} = p_{q2}$ then the number of occurrences of p_{q1} is 2. This value is then inserted in the corresponding cell for the p_{q1}. Figure 4 shows an example of a query vector.

p_{q1}	p_{q3}	...	p_{qj}	...	p_{qt}
2	1	...	2	...	0

Fig. 4. Example of the query vector

Subsequently, each pair of the vector *vq* is searched in the index (matrix *Mec*) in order to obtain the number of times it is found in each *BPs* stored in the repository. This number is then multiplied by the corresponding value in the vector vq, and the resulting value is inserted in a new matrix named "query matrix" (*Mq*) where rows are the *BPs* of the repository and columns are the node pairs of the query vector vq (See Figure 5)

	p_{q1}	p_{q3}	...	p_{qj}	...	p_{qt}	ec-sim
BP_1	0	0	...	4	...	0	4
BP_2	6	0	...	0	...	0	6
...	...	...	...	...	...	...	...
BP_i	4	0	...	6	...	0	10
...	...	...	...	..	...	...	...
BP_n	2	5	...	0	...	0	7

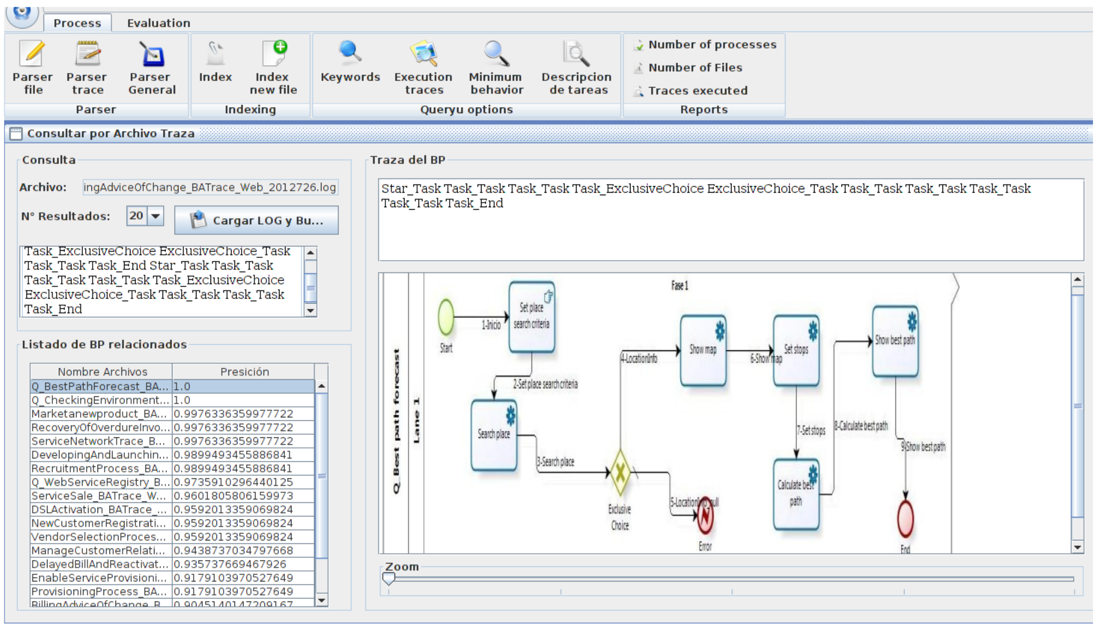
Fig. 5. Example of the query matrix (Mq) plus the similarity for each BP

For example, the vector query of Figure 2 and the execution cases matrix of Figure 1, suppose that $p_{q1} = p1$, $p_{q3} = p_2$, $p_{qj} = p_j$ *and* $p_{qt} = p_k$. The pair p_{qj} with an occurrence of 2 in the vector query *vq* is found three times in the execution cases matrix, hence by multiplying those values we get 6; this value is inserted on the cell (*i; j*) of the query matrix *Mq*. Finally, in order to rank the BPs of the repository, the values of each row are added obtaining a value of execution cases similarity (*ec-sim*) for each *BPs*. Accordingly, the BPs are ranked from the greatest value to the lowest one. The complete resulting query matrix of the example is presented in Figure 5 where the resulting ranking is r = $\{BP_i(10), BP_n(7), BP_2(6), BP_1(4)\dots\}$.

3.3 A tool for implementing Trazas BP

The tool that implements TrazasBP was developed in Java and integrates a user centered interface. This tool incorporates some usability criteria defined by objective and subjective attributes. Among the objective attributes, the tool integrates: ease of learning and memorization, efficiency, effectiveness, operability and ease of understanding, equally. Additionally, subjective attributes (oriented to user satisfaction) supported in the tool are: accessibility, functionality, usefulness and credibility. The tool includes a simple interface with panels

ISSN 2395-8618



Consulta

Archivo: ingAdviceOfChange_BATrace_Web_2012726.log

N° Resultados: 20

Task_ExclusiveChoice ExclusiveChoice_Task Task_Task Task_End Star_Task Task_Task Task_Task Task_Task Task_ExclusiveChoice ExclusiveChoice_Task Task_Task Task_Task Task_End

Listado de BP relacionados

Nombre Archivos	Presición
Q_BestPathForecast_BA...	1.0
Q_CheckingEnvironment...	1.0
Marketanewproduct_BA...	0.9976336359977722
RecoveryOfOverdureInvo...	0.9976336359977722
ServiceNetworkTrace_B...	0.9976336359977722
DevelopingAndLaunchin...	0.9899493455886841
RecruitmentProcess_BA...	0.9899493455886841
Q_WebServiceRegistry_B...	0.9735910296440125
ServiceSale_BATrace_W...	0.9601805806159973
DSLActivation_BATrace_...	0.9592013359069824
NewCustomerRegistrati...	0.9592013359069824
VendorSelectionProces...	0.9592013359069824
ManageCustomerRelati...	0.9438737034797668
DelayedBillAndReactivat...	0.935737669467926
EnableServiceProvisioni...	0.9179103970527649
ProvisioningProcess_BA...	0.9179103970527649
BillingAdviceOfChange_B...	0.9045140147209167

Traza del BP

Star_Task Task_Task Task_Task Task_ExclusiveChoice ExclusiveChoice_Task Task_Task Task_Task Task_Task Task_Task Task_End

Fig. 6. User interfaces for performing queries

containing the functionalities of the model. Equally, the user may choose a BP model from the result list in order to visualize and thus check the validity of the query.

Figure 6 shows the results of one example query, in this figure results are displayed (red square) when the user performs the query. The results contain the more relevant BP models according to the similarity between the query and the BP in the repository.

IV. Evaluation and Results

Because For the experimental evaluation, TrazasBP was used for generating rankings of 20 BP according to the similarity with the Query BP. This procedure evaluates the relevance of the results retrieved in each search. The evaluation was performed using the measured widely used for evaluating information retrieval systems: Graded Precision, graded recall and F-measure.

Relevance evaluation of results in TrazasBP includes two phases: The first one evaluates relevance and quality of ranking, in order to find the best query option between: Execution case, Minimal behavior, and Log-file. The second phase compares results obtained using TrazasBP with the results of the manual evaluation performed on a closed test set, which was previously described in [21]. This closed test set was created collaborative by 59 experts. Moreover, the ranking generated by evaluators and the ranking automatically generated using the TrazasBP were compared using the measure A(Rq) presented in [22]. A(Rq) measure was used to determine the degree of coincidence of the position of each BP in each one of the rankings generated by each request.

Figure 7 shows results of the first phase, where for each querying option the Graded Precision (Gp), graded recall(Gr) and F-measure (Gf) are calculated. Graded precision reached values between 81% and 90% which means that the present approach is less likely to retrieve non-relevant BPs (i.e. false positives). Nevertheless, the lower values of recall from 19% to 28% demonstrate that the approach doesn't retrieve a high number of relevant BPs (i.e. false negatives). With regards to the F-measure, the approach obtained values from 30% to 42% for the different query options showing acceptable values of harmony between the precision and recall measures. Results of phase one show that query option based on Log-file achieved the best results, this is because each log file integrated many execution cases of the same BP, which extends the possibility for finding BP with similar execution cases in the repository. Consequently, the query option based on log files was selected for phase two.

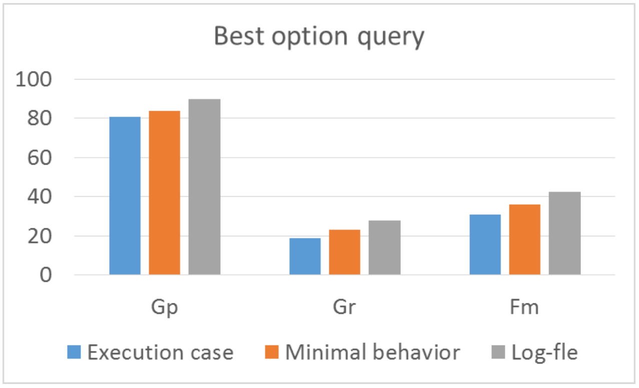


Fig. 7. Results of querying options comparison

ISSN 2395-8618

Results of phase two are shown in Figure 8. These results show that Trazas Bp achieved a 94% of Gp, therefore search results are precise and keep high similarity with the ranking generated by human evaluators. In other words, TrazasBP retrieves most of the BP that human evaluators considered as relevant for each query.

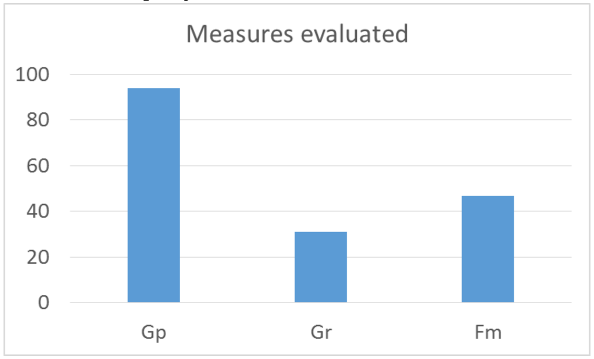


Fig. 8. Results of the phase 2

With regards to Graded precision, TrazasBP reached a value of 31%. This is due to TrazasBP generated rankings limited to 20 results; and it left aside other BPs relevant for the query. The results for graded F-measure evidenced harmony in the results of Gp and Gr. The average value of Gf is 47% which indicates that classifications generated by TrazasBP presented high similarity with the human generated ranking described in [21].

Figure 9 depicts the level of agreement A(Rq) between the ideal ranking generated by evaluators and the automatic classification generated using TrazasBP. Note that for each query the proposed approach generated classifications that match considerably with those generated by experts (ideal classifications). For example, in query 1 (Q1) the similarity of classification for the proposed method reached 85%. Finally, in the classification of global similarity (considering all the queries) TrazasBP reaches 81%. This result indicates an increase in quality of the generated ranking when Log files are used as query element. TrazasBP retrieves the most relevant list for each query and avoids retrieving no-relevant BP

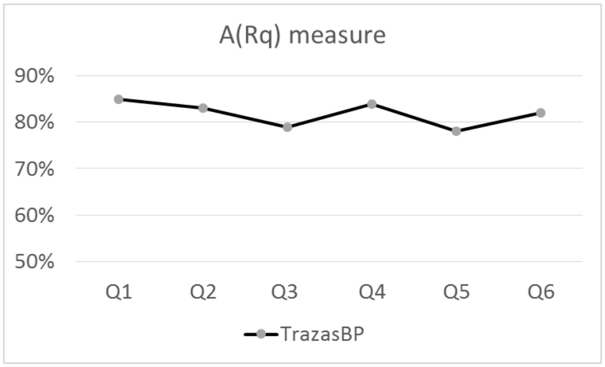


Fig. 9. Ranking concordance for each query (A(Rq))

V. Conclusion and Future Works

This paper presents TrazasBP, a framework for BP indexing and searching based on execution cases. TrazasBP indexes BPs based on execution cases retrieved from log files. Additionally, it considers not only textual information of BP elements but also causal dependence between BP elements. TrazasBP was evaluated in two phases: The first one evaluated relevance and quality of ranking using different querying options, and found as the best ranking option the one based on log-files. During the second phase, the present approach was compared with results obtained by human experts. Results obtained in this phase allow evidence that TrazasBP generates rankings of results with high similarity to the rankings generated by humans.

Experimental evaluation evidenced high values precision (90%) for different query options. Additionally, the F-measure reached values around 42% which is an acceptable value for the relation between precision and recall. Equally, When comparing TrazasBP with a closet test created by human experts, the Graded precision reached 94% which shows that the ranking generated with TrazasBP is highly similar to the ranking generated by human experts. Due to TrazasBP low computational cost, it may be effectively used as indexing mechanism and may precede other expensive algorithms during BPs similarity calculation since it reduces the search space. Additionally, TrazasBP approach can be extended by adding new query options.

Future work includes incorporating new search options: i) semantic options by adding domain ontologies that represent user queries. ii) multimodal options which consider structural, behavioral, and linguistic information in one search space. Equally, future work will include integration of clustering algorithms (like K-means, Clicks, Start, and C–means) to the model and compare the created groups with other results reported in the state of the art. On the other hand, it is planned to develop an automatic evaluation module that generates graphs and relevance measures. Finally, evaluation will be expanded by applying new measures for the BP search.

References

[1] H. A. Reijers, R. S. Mans, and R. a. van der Toorn, “Improved model management with aggregated business process models,” *Data Knowl. Eng.*, vol. 68, no. 2, pp. 221–243, Feb. 2009.

[2] F. M. Maggi, M. Dumas, and F. B. Kessler, “Predictive Monitoring of Business Processes,” In Proc. *26th International Conference, CAiSE 2014*, Thessaloniki, Greece, pp.457-472.

[3] F. Rahimi, C. Møller, and L. Hvam, “Business process management and IT management: The missing integration,” *Int. J. Inf. Manage.*, vol. 36, no. 1, pp. 142–154, Feb. 2016.

[4] I. Khodyrev and S. Popova, “Discrete Modeling and Simulation of Business Processes Using Event Logs,” *Procedia Comput. Sci.*, vol. 29, pp. 322–331, 2014.

[5] W. M. P. van der Aalst, H. A. Reijers, A. J. M. M. Weijters, B. F. van Dongen, A. K. Alves de Medeiros, M. Song, and H. M. W. Verbeek,

ISSN 2395-8618

"Business process mining: An industrial application," *Inf. Syst.*, vol. 32, no. 5, pp. 713–732, 2007.

[6] J. De Weerdt, M. De Backer, J. Vanthienen, and B. Baesens, "A multi-dimensional quality assessment of state-of-the-art process discovery algorithms using real-life event logs," *Inf. Syst.*, vol. 37, no. 7, pp. 654–676, Nov. 2012.

[7] I. Bider, K. Gaaloul, J. Krogstie, S. Nurcan, H. A. Proper, R. Schmidt, and P. Soffer, "Enterprise, business-process and information systems modeling," in *Lecture Notes in Business Information Processing*, 2014, vol. 175.

[8] J. Y. In P. Maglio, M. Weske and and M. Fantinato, "Discovering business process similarities: An empirical study with sap best practice business processes," in Proc *International Conference on Service-Oriented Computing*, pp. 515-526, 2010.

[9] A. K. Alves de Medeiros, W. M. P. Van der Aalst, and A. J. M. M. Weijters, "Quantifying process equivalence based on observed behavior," *Data Knowl. Eng.*, vol. 64, no. 1, pp. 55–74, 2008.

[10] R. Dijkman, M. Dumas, and L. García-Bañuelos, "Graph matching algorithms for business process model similarity search," *Bus. Process Manag.*, vol. Business P, pp. 48–63, 2009.

[11] Z. Yan, R. Dijkman, and P. Grefen, "Fast business process similarity search with feature-based similarity estimation," in Proc *OTM Confederated International Conferences*, 2010, pp. 60–77.

[12] S. Goedertier, D. Martens, J. Vanthienen, and B. Baesens, "Robust Process Discovery with Artificial Negative Events," *J. Mach. Learn. Res.*, vol. 10, pp. 1305–1340, 2009.

[13] M. Segatto, S. I. D. De Pádua, and D. P. Martinelli, "Business process management: a systemic approach?," *Bus. Process Manag. J.*, vol. 19, no. 4, pp. 698–714, 2013.

[14] J. Bae, L. Liu, J. Caverlee, L.-J. Zhang, and H. Bae, "Development of Distance Measures for Process Mining, Discovery and Integration," *Int. J. Web Serv. Res.*, vol. 4, no. 4, pp. 1–17, 2007.

[15] M. Weidlich, A. Polyvyanyy, J. Mendling, and M. Weske, "Causal behavioural profiles - Efficient computation, applications, and evaluation," in *Fundamenta Informaticae*, 2011, vol. 113, no. 3–4, pp. 399–435.

[16] R. Dijkman, M. Dumas, B. van Dongen, R. Käärik, and J. Mendling, "Similarity of business process models: Metrics and evaluation," *Inf. Syst.*, vol. 36, no. 2, pp. 498–516, Apr. 2011.

[17] H. Zha, J. Wang, L. Wen, C. Wang, and J. Sun, "A workflow net similarity measure based on transition adjacency relations," *Comput. Ind.*, vol. 61, no. 5, pp. 463–471, 2010.

[18] A. Wombacher and M. Rozie, "Evaluation of workflow similarity measures in service discovery," *Serv. Oriented Electron. Commer.*, vol. 7, no. 26, pp. 51–71, 2006.

[19] K. Gerke, J. Cardoso, and A. Claus, "Measuring the compliance of processes with reference models," in *Proc OTM Confederated International Conferences*, 2009, pp. 76–93.

[20] J. Wang, T. He, L. Wen, N. Wu, A. H. M. Ter Hofstede, and J. Su, "A behavioral similarity measure between labeled Petri nets based on principal transition sequences (short paper)," in *Proc OTM Confederated International Conferences*, 2010, pp. 394–401.

[21] J. C. Corrales, C. Cobos, L. K. Wives, and L. Thom, "Collaborative Evaluation to Build Closed Repositories on Business Process Models," in Proc *ICEIS*, 2014, pp. 311–318.

[22] M. Guentert, M. Kunze, and M. Weske, "Evaluation Measures for Similarity Search Results in Process Model Repositories," pp. 214–227, 2012.