Clustering Based Compliance Methodology

Nour Damer[1] , Mieke Jans[1], Benoit Depaire[1] and Koen Vanhoof[1],

[1] Hasselt University, Agoralaan Building D, 3590 Diepenbeek, Belgium

{Nour.Damer, Mieke.Jans, Benoit.Depaire, Koen.Vanhoof} @uhasselt.be

Abstract. The general idea of compliance methodologies is to ensure that business processes are executed as they are designed. Compliance methodologies presented in previous work assume one representation for the actual behavior. However, if this representation is very general, the result of the compliance method will not be accurate enough to measure the performance. In this paper, we present a new methodology to check the compliance of unstructured processes based on clustering, taking into account different sub-patterns of actual behavior. We believe that clustering log files generate more representative and accurate models and hence a better start to measure compliance between the actual and the prescribed behavior. In the paper a case study is presented and its results discussed

Keywords: Business Process Compliance, Process Mining, Sequence Clustering.

1 Introduction

Businesses, generally talking, work by executing a set of prescribed processes derived from the organization objectives. During the run-time, these prescribed processes are used as controls to monitor the actual execution of processes. The procedure of ensuring that actual behavior of processes is in accordance with the prescribed processes is referred to as process compliance. To check the compliance of a business process, the actual behavior should be described first to be compared with the prescribed process. This can be obtained by a variety of process modeling techniques as process mining which showed promising results during the last decade [1].

Process mining can be seen as a subfield of data mining where the input data is data of the business processes. It is based on mining the data recorded in log files to extract information about the business processes as they are being executed. Its results were promising when applied to structured processes such as workflows [2]. However, execution processes in real life leads to high diversity of behavior. Mining such processes using traditional techniques generate spaghetti-like models which are very complex and difficult to understand. A proposed solution for this problem was to discard unusual behavior [3]. Although this seems to be a good solution, it does not work properly when we are interested in unusual cases. To overcome these problems, clustering approach was proposed. Clustering based algorithms divide the log file into

clusters of similar cases. Clustering provides a more accurate representation for the similar cases. On the other hand, some clustering algorithms, such as sequence clustering used in this study, support discarding the unusual behavior which can affect assigning the cases into the clusters negatively. Research indicated that sequence clustering with appropriate preprocessing can be the first approach to process mining [4]. It is able to deal with this heterogeneous character and that is why we believe it is a good start.

In this work, we propose a new methodology to check the compliance of unstructured processes based on clustering. By unstructured we mean processes which have a high diversity of behavior. We believe that clustering log files before modeling generate more representative and accurate models. Accordingly, we thought of a methodology that divide the whole log file into clusters of similar sequences before checking for compliance.

The remaining paper is structured as follows: The Sequence Clustering algorithm used in this work is explained in Section 2. The Process Compliance concept is discussed in Section 3. Our proposed methodology based on Clustering is presented in Section 4. In section 5 the proposed methodology was applied to a real life case study. Finally the conclusion is given in Section 6.

2 Sequence Clustering in Process Mining

Recently, the use of clustering techniques in process mining has become more significant. Some studies as the one in [2] make use of the k-means clustering algorithm to cluster workflow traces where each trace is seen as a point of a properly identified space of features. A similar technique known as the Disjunctive Workflow Schema (DWS) was proposed in [5]. It is a hierarchical clustering based approach which splits the log into clusters iteratively using the K-means method. The work of [6] is also a hierarchical clustering based approach but it first transforms the business process models into vector models according to their structures. Trace clustering [9] is another clustering algorithm which is based on a set of profiles, each measuring a number of features for each case from a specific perspective. Recent work in trace clustering includes the use of an edit distance between sequences [7].

Like the clustering techniques described above, sequence clustering provides the means of partitioning a number of cases into a set of clusters of similar cases [8]. However, the input for sequence clustering is the sequences itself, not the features extracted from these sequences. Sequence clustering was first developed in bioinformatics to group large protein datasets into different families [9]. Later on, the idea was applied to process mining to discover the typical behavior of different processes or to discover the different behaviors within a single process [4].

Normally, the behavior of one business process differs from one case to another. Real life data usually represents different behaviors because of its dynamic nature. The result of this heterogeneous character is a set of different behaviors and accordingly different workflow sequences. This diversity creates the need to study the different behaviors separately. For this purpose, sequence clustering algorithm

was implemented to study the cases of a log file according to the order of performing the recorded activities, i.e their workflow sequences.

Initially, the sequence clustering algorithm generates k clusters where each cluster is associated with a probabilistic model which is usually a simple Markov chain model. When both clusters and their corresponding Markov chains are generated, each individual sequence in the input log file is assigned to one of the generated clusters. To assign a given sequence, the probability of each cluster producing this sequence is computed first. The sequence is assigned to the cluster which has the highest probability to produce this specific sequence. Later on, these steps are repeated an iterative Expectation-Maximization procedure to re-estimate the models. The sequences assigned to each sequence are used to re-estimate the Markov chain of that cluster. This produces more accurate models to be used to assign the sequences again in the next iteration. By repeating the procedure again and again, the models approach to be more stable Markov chains which no longer change. The final output Markov chains are the models that represent the behavior of its associated cluster [4].

The probabilistic model associated with each cluster can accommodate several behaviors, so that, each sequence in the log is assigned to one of the clusters. This could be a double-edged sword. From one side, sequence clustering is considered to be robust to noise. On the other hand, assigning unique sequences will affect the probabilistic models negatively. [3,4]. To solve this problem, some preprocessing can be done before running the algorithm.

Preprocessing filters the undesired behavior that affects the probabilistic models negatively. The algorithm provides different preprocessing options to filter undesired events as well as undesired sequences. The selection of one or more preprocessing step depends on the application we are analyzing. For example, in case of analyzing the typical behavior, we can remove the infrequent events or unique sequences. For more details about the usually preprocessing steps performed, readers may refer to [4].

Research indicated that sequence clustering with appropriate preprocessing can be used as the first approach to process mining [4]. It is robust to noise and able to deal with very large volume of data with different behaviors. It is useful to discover the behavior of different processes in the log file and divide them into clusters. Also it can be used to discover the different behaviors of one process and then visualize the results [3,8].

3 Process Compliance

From business point of view, executing processes efficiently leads to better performance which in turn means higher profit. Processes are designed in an early stage in a way to achieve the organization objectives. During the run-time, the challenge is to perform the processes as designed. This is referred to as business process compliance [10].

Compliance is concerned with ensuring that business processes, operations and practice are in accordance with the prescribed controls during the process design. Although compliance is not a new topic in research literature, it started receiving

more attention by businesses as a way to improve their business processes. Industry reports [4] indicate that up to 80% of companies said they expected to reap business benefits from improving their compliance regimens. The market value for compliance related software and services are estimated over $32 billion in 2008 [11,12].

Checking compliance can be viewed from three different perspectives: corrective, preventative, and detective. The last perspective is the concern of this work. With compliance detective we aim to examine how compliant the actual behavior of a process is with regard to the controls. In our case, prescribed processes are the controls to be followed, so that we will use the two terms interchangeably.

4 Clustering Based Compliance Methodology

The compliance methodology described in this work is based on clustering the log file before checking the compliance. We believe that clustering provide a better representation of the actual behavior. The idea is to divide the log file into a set of clusters of similar cases and then compute a compliance degree between each cluster and the designed model. Later on, we sum up the calculated values for each cluster into one value.

Going more into details, we can say that there are three main sub-processes to be performed before finding out the final result. These are:

1. Clustering the log file using a clustering algorithm
2. Applying a compliance methodology for each cluster
3. Calibrating the resulted compliance degrees with their occurrence in the log file

Regarding these sub-processes, some issues should be considered such as which clustering algorithm and compliance methodology are best suitable to be used. In our case we prefer to use the sequence clustering algorithm since we are mainly concerned with the following order of executing the events. The algorithm gives a set of clusters of similar sequences as described in section 1.2. We use these cases associated with each cluster to generate a behavioral model for that cluster. The generated model is then used as an input to compute the compliance for each cluster separately.

As a compliance methodology, we use the Compliance by Design presented in [11]. This methodology is used to measure the distance between the controls and the actual behavior model quantitatively. Later on, the measured compliance degrees are calibrated in the third step using the frequencies of execution similar to the work of [10]. Finally, the values of compliance for the different clusters are then summed up in one value to compare with the compliance degrees of the whole log file before clustering.

5 Case Study: Procurement Process

The proposed methodology was applied to a real life set data from the business world. The cooperative organization is an international financial services provider. It was ranked as one of the top 20 financial organizations in Europe.

The input data file is an example of large log files with high diversity of behavior. It was derived from the procurement process cycle configured in the SAP system. The cycle starts by creating a purchase order and ends with the payment of associated invoices. Each case represents one purchase order item line. All activities preformed on a particular case make the audit trial of that case. The procedure of process selection and data preparation was described thoroughly in the work of [13] where a framework for internal fraud risk reduction was introduced. Although this framework does not explicitly focus on compliance, it also aims at checking whether procedures are followed or not. In this paper we broaden our focus from fraud risk reduction to compliance, but the steps of process selection and data preparation can however still be adopted from [13]. It goes beyond the scope of this paper to represent the details of the whole procedure, so interested readers may refer to the original work in [13].

We have divided our work into four stages: deducing controls from the process design, applying the compliance by design methodology to the log file as a whole, applying the clustering based methodology and finally compare the results.

5.1 Deducing Controls from the Process Design

In this stage we prepare the controls and represent them in a formal language. For this purpose, we refer to the prescribed model, shown in Figure 1, which was designed by the case company for the procurement process. We convert this model into a set of controls using the Formal Contract Language (FCL) [14, 15]. The output of this stage is a set of rules each representing one control.

Figure 1 represents the procurement process at our case company. At the beginning, the purchase order (PO) is created with its item line(s) and signed before being released. In some cases the PO can be released without an additional signature (sign). After a release, Goods and invoice are received. Finally, after both goods and invoice are received the payment can be made. However, in some cases no Goods are received such as services orders. Often, the item line will be changed between the creation, Sign and Release activities. It is also possible to change the line after it was released but in such cases a new sign and release need to be triggered.

FCL is a formal control modeling language which has proved high ability to reason with violations. Normally, business processes are deployed in a dynamic environment which affects the execution of processes. In some cases, it is not possible to execute the processes as designed. However, a good process design prescribes how to recover from the resulting violations. In other cases the prescribed processes are themselves a subject to exceptions. FCL has the ability to deal with such cases because of its combination of an efficient non-monotonic formalism (defeasible logic) and deontic logic of violations [16].

A rule in FCL is an expression of the form r:A1,..., An ➔ B, where r is a unique name for each rule, A1,..., An are the premises, and B is the conclusion of the rule.

Both premises and conclusion are propositions of the logic. The propositions of the logic are built from a set of propositions connected by four operators. In this study, we have used the violation/reparation operator denoted by ⊗. In FCL, saying that A⊗ B means that in case A is not fulfilled then B has to be fulfilled [14, 15].

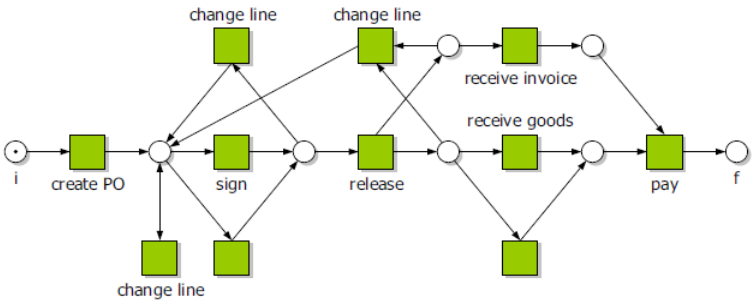

Fig. 1. Prescribed Model of Procurement Process in Petri Net

The current definition of FCL does not completely support representing our controls. The activity Change line can occur after different activities as discussed earlier. Its occurrence is considered as a sub-ideal state. However, when it occurs, it should be followed by one of four activities, Sign, Release, Receive Invoice (IR), or Receive Goods (GR) with no preferences. One solution was to study the occurrence of this activity between the other activities, but creates more complicated states. Another solution was to use four different rules to represent each activity. This is also inconvenient because it means that the model should provide four different paths from Change line to each of the four activities to have a suitable degree of compliance.

As a solution we proposed using a new operator, the Or operator, to represent no preferences cases such as the Change line in our case study. We use the Or operator in one rule to represent that if one or more of the four activities occurs after changing an item line then it is an ideal state. Controls deduced from the prescribed model in our case are expressed in six unique rules:

- r1: Create PO➔ Sign ⊗ (Release, Change line)
- r2: Sign ➔ Release ⊗ Change line
- r3: Release ➔ Receive Goods ⊗ (Receive Invoice, Change line)
- r4: GR ➔ IR ⊗ (Pay, Change line)
- r5: IR ➔ Pay ⊗ Change line
- r6: Change line ➔ Sign Or Release Or IR Or GR

These rules are still not finalized and need to be converted into a form such that it is comparable to the process model. For this purpose, we used the ideal semantics concept which define four states of idealness; ideal, sub-ideal, and non-ideal. The states of idealness reflect how well a process model complies with the controls. A state is said to be ideal if the execution behavior is fully compliant with the control rule. A sub-ideal state is a state where there are some violations, but these could be recovered. Both states are compliant; however sub-ideal states are expected to provide sub-optimal performance. A state is considered as non-ideal if it violates a control

without being repaired [16]. Our states of idealness, sub-idealness, and non-idealness are shown in Table 1.

Table 1: States of Idealness of Controls

	Rule	Ideal state	Sub-ideal state	Non-ideal state
R1	Create PO => Sign ⊗ (Release, Change line)	<Create PO, Sign>	<Create PO, Release> <Create PO, Change line>	<Create PO, IR>
R2	Sign => Release ⊗ Change line	<Sign, Release>	<Sign, Change line>	
R3	Release => GR ⊗ (IR, Change line)	<Release, GR>	<Release, IR> <Release, Change line>	<Release, Sign> <Release, Pay>
R4	GR => IR ⊗ (Pay, Change line)	<GR, IR>	<GR, Pay> <GR, Change line>	<GR, Sign>
R5	IR => Pay ⊗ Change line	<IR, Pay>	<IR, Change line>	<IR, Pay, IR>
R6	Change line => Sign Or Release Or IR Or GR	<Change line, Sign> <Change line, Release> <Change line, IR> <Change line, GR>		

5.2 Applying Compliance by Design Methodology

To apply any compliance methodology, we need first to prepare both sides: the controls we want to test and the process model as it is executed. Controls have already been prepared and presented in a comparable form in the previous stage. Like in [13] we use a process discovery technique in the ProM framework namely the FuzzyMiner to present the process as executed.

The output model, shown in Figure 2, was produced using the same preprocessed data used as an input to the sequence clustering algorithm. With both sides defined, the next step is to measure the compliance between the actual behavior and the controls. We prefer to use a quantitative method, so that it will be easier to compare results later on. For this purpose, we used the compliance by design methodology presented in [11] to measure the distance between the actual behavior model and the controls. The idea is to measure how well the behavioral model supports the execution of the different sates of idealness. In this work, we will refer to the support of ideal, sub-ideal, and non-ideal sequences as ideal, sub-ideal, and non-ideal compliance degree respectively. Each measurement indicates whether the state is fully or partially supported in the model. The measurement was then calibrated to include the frequencies of execution in the log file.

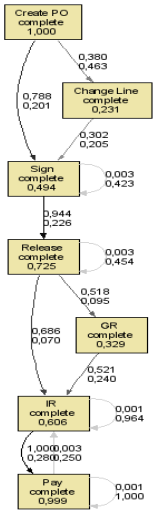

Fig. 2. Actual Behavior Model

The calibrated degree given in Table 2 is the product of the compliance degree and its frequency. The idea behind using the frequencies is to calibrate the degrees to have more accurate values. When a state has the degree (1) of compliance, this means that the model completely supports its execution. However, this does not consider how many cases are supported out of the total number of cases given in the case study. For non-ideal states, the model represents only one state which is associated with Rule 5: <IR, Pay, IR> with 27 frequencies so that the non-ideal compliance degree is 27.

Table 2: Ideal and sub-ideal Compliance Degree

	Ideal state	Calibrated Degree	Sub-ideal state	Calibrated Degree
R1	<Create PO, Sign>	1 * 6602 = 6602	<Create PO, Release>	0 * 256 = 0
			<Create PO, Change line>	1 * 2863 = 2863
R2	<Sign, Release>	1 * 8779 = 8779	<Sign, Change line>	0 * 38 = 0
R3	<Release, GR>	1 * 4103 = 4103	<Release, IR>	1 * 4867 = 4867
			<Release, Change line>	0 * 845 = 0
R4	<GR, IR>	1 * 4219 = 4219	<GR, Pay>	0 * 170 = 0
			<GR, Change line>	0 * 15 = 0
R5	<IR, Pay>	1 * 9578 = 9578	<IR, Change line>	0 * 44 = 0
R6	<Change line, Sign>	1 * 2300 = 2300		
	<Change line, Release>	0 * 717 = 0		
	<Change line, IR>	0 * 641 = 0		
	<Change line, GR>	0 * 147 = 0		
	Sum	**35581**	**Sum**	**7730**

5.3 Applying Clustering Based Compliance Methodology

As described in Section 3, our methodology is to check the process compliance. Unlike other compliance methodologies, by representing the actual executed process by means of K models instead of only one model. The K models are the output of a preprocessing step where we apply sequence clustering. We suggest to divide the log file into a set of clusters of similar sequences. As a start, we used the sequence clustering algorithm to cluster the log file. As a result, we had four clusters with a log file associated with each cluster.

Now, let us look more into detail, we have a log file of 10,000 different cases. The average number of events per case is six while the maximum number is 202. The preprocessing options available within the sequence clustering tool include ignoring events and sequences with low support. In our case, we have adjusted the minimum number of sequence occurrence from default to block the uncommon cases to achieve better representation.

The algorithm was run 20 times with different predefined number of clusters between two and eight. Preprocessing was considered in 65% of the times. Comparing results using the conformance checker tool under ProM framework shows

that the best output was given by four clusters after tuning the minimum number of sequence occurrence to five. The result of this clustering step is displayed in Figure 3. Approximately, 2% of the cases were discarded after preprocessing.

After preparing the models, and given the states in Table 1, we compute the compliance of each cluster using the same method described in Section 6.2. Compliance degrees for all assigned clusters are given in Table 3.

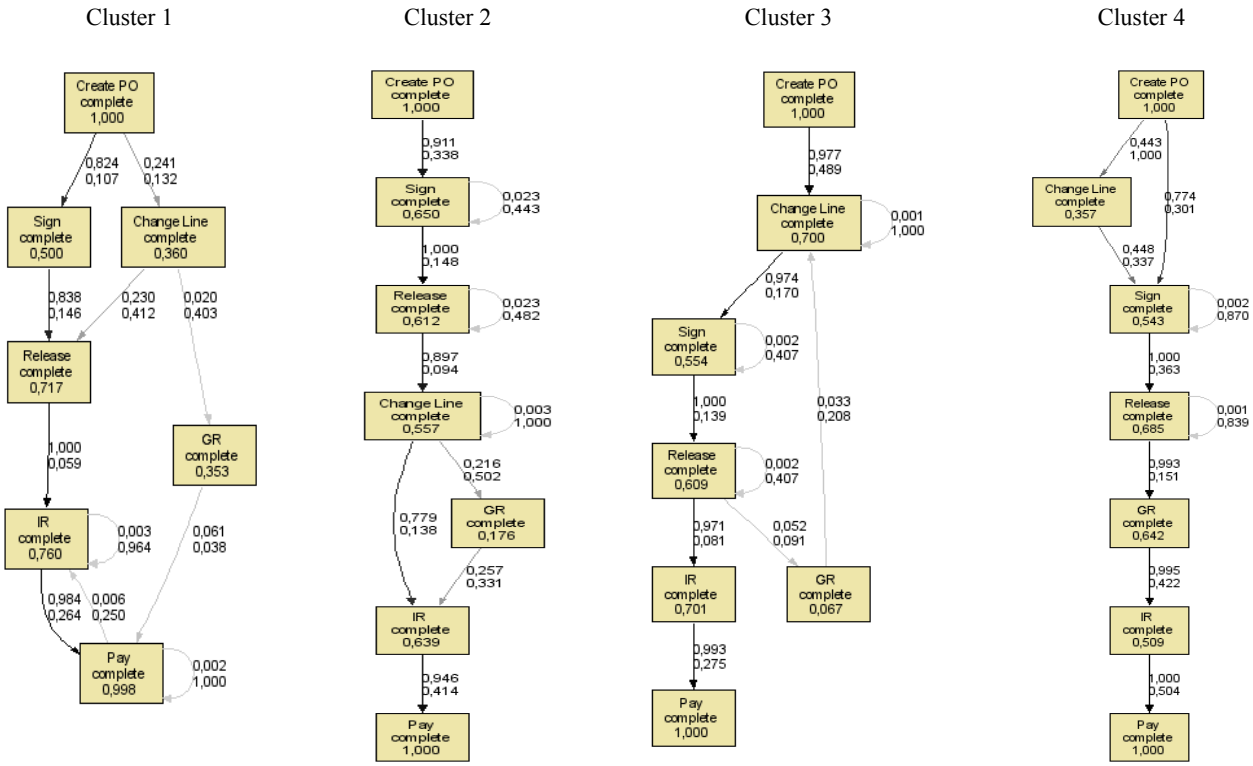

Fig. 3. Fuzzy Miner models produced by the Fuzzy Miner algorithm to model each of the four clusters generated using the Sequence Clustering Algorithm

Table 3: Compliance Degree for each Cluster

Sub-Ideal State	Compliance Degree	Frequency	Calibrated Degree
Cluster 1	11179	5052	27
Cluster 2	3365	785	0
Cluster 3	2028	1327	0
Cluster 4	20326	1449	0
Sum	36898	8613	27

The results presented in Table 4 show that for both ideal and sub-ideal states, the compliance degrees are higher after clustering the log file. For non-ideal states, only one of the states <IR, Pay, IR> is represented in tow models, the general model and cluster 1 model. This leads to have the same degree of non-ideal before and after clustering. Results show that clustering has a positive impact on the overall compliance. Clustering results in a higher compliance degree on ideal and sub-ideal states, and an equal compliance degree with non-ideal states.

Table 4: Compliance Degree for each Cluster

Sub-Ideal State	Compliance Degree	Frequency	Calibrated Degree
Traditional methodology	35581	7730	27
Clustering based methodology	36898	8613	27

7 Conclusion

In this paper, we present a Clustering based Compliance methodology to measure the degree of compliance between control objectives and business process models of the run time. Since we are concern with the sequence of occurrence, we use the sequence clustering algorithm to cluster similar sequences. Sequence clustering is used also because of its ability to deal with noise and heterogeneous character captured in reality.

The proposed methodology was applied to a procurement process from the real life. Results show that the overall compliance degree was enhanced after clustering the log file taking into consideration the frequencies of execution. The overall compliance after clustering was higher for both ideal and sub-ideal states. For non-ideal states, the compliance is exactly the same. As a future work, we aim to study some modeling techniques other than the Fuzzy Miner used in this study. We believe that the models extracted play a role in the computation because as much as the model is over-generalized, as much as it represents more behavior and leads to higher compliance degrees with both ideal and non-ideal stated. On the other hand, this will affect the non-ideal state negatively because a higher degree of a non-ideal state means less optimality.

References

1. Van der Aalst, W.M.P., Weijters, A.J.M.M.: Process mining: a research agenda. Computers in Industry 53, 231–244 (2004)
2. Greco, G., Guzzo, A., Pontieri, L., Saccà, D.: Mining Expressive Process Models by Clustering Workflow Traces, In: 8th Pacific-Asia Conference (PAKDD 2004), LNCS 3056, Springer, (2004)

3. Veiga, G., Ferreira, D.: Understanding Spaghetti Models with Sequence Clustering for ProM, In: Business Processing Intelligence 2009 Workshop, (2009)
4. Ferreira, D.: Applied sequence clustering techniques for process mining. In Cardoso, J., van der Aalst, W., eds.: Handbook of Research on Business Process Modeling. pp. 492-513, Information Science Reference, IGI Global (2009)
5. De Medeiros, A.K.A., Guzzo, A., Greco, G., van der Aalst, W.M.P., Weijters, A.J.M.M., van Dongen, B.F., Saccμa, D.: Process mining based on clustering: A quest for precision. In ter Hofstede, A.H.M., Benatallah, B., Paik, H.Y., eds.: Business Process Management Workshops. Volume 4928 of Lecture Notes in Computer Science., pp. 17-29. Springer (2007)
6. Jung, J., Bae, J., Liu, L.: Hierarchical Clustering of Business Process Models, International Journal of Innovative Computing, Information and Control, Volume 5, Number 12, (December 2009)
7. Bose, R.P.J.C., van der Aalst, W.M.P.: Context aware trace clustering: Towards improving process mining results. In: SDM, SIAM 401-412 (2009)
8. Ferreira, D., Zacarias, M., Malheiros, M., Ferreira, P.: Approaching process mining with sequence clustering: Experiments and findings. In: Proceedings of the 5th International Conference on Business Process Management (BPM 2007). Volume LNCS 4714, Springer 360-374 (2007)
9. Chen, Y., Reilly, K., Sprague, A., Guan, Z.: SEQOPTICS: a protein sequence clustering system, In: BMC Bioinformatics, 7 (Suppl 4):S10, (2006)
10. Lu, R., Sadiq, S., Governatori, G.: Measurement of Compliance Distance in Business Work Practice. IS Management 25, 4: 344-355 (2008)
11. Lu, R., Sadiq, S., Governatori, G.: Compliance Aware Business Process Design. In: the 3rd International Workshop on Business Process Design (BPD'07). In conjunction with the 5th International Conference on Business Process Management, Springer Berlin / Heidelberg LNCS Volume 4928/2008, Pg. 120-131. (2008)
12. Sadiq, S., Governatori, G.: A Methodological Framework for Aligning Business Processes and Regulatory Compliance. In Brocke, J., Rosemann, M. (eds.) Handbook of Business Process Management. Introduction, Methods and Information Systems Series: International Handbooks on Information Systems. Springer (2010)
13. Jans, M.: A Framework for Internal Fraud Risk Reduction: The IFR² Framework, PhD thesis, University of Hasselt, Diepenbeek, Belgium (2009)
14. Governatori, G.: Representing Business Contracts in RuleML. International Journal of Cooperative Information Systems 14(2–3), 181–216 (2005)
15. Governatori, G., Milosevic, Z.: A Formal Analysis of a Business Contract Language. International Journal of Cooperative Information Systems 15(4), 659–685 (2006)
16. Governatori, G., Sadiq, S.: The Journey to Business Process Compliance. In Cardoso, J., Van der Aalst, W. (eds) Handbook of Research on Business Process Modeling, pp. 426-454, Information Science Reference, IGI Global (2009)