Leveraging on Responsibilities for Optimal Process Alignment Computation

Matteo Baldoni[1], Cristina Baroglio[1], Elisa Marengo[1,*] and Roberto Micalizio[1]

[1]University of Turin, Computer Science Department – Turin, Italy

Abstract

Process alignment aims at relating an actual execution, captured as a log trace, to a process run of a model. Besides considering the control flow aspect, in this paper we propose to leverage on contextual information expressed in terms of responsibilities. This allows us to select, among the possible alignments, those where the fewest number of responsibilities is neglected. Additionally, we leverage on responsibilities to evaluate the deviations occurring between a process run and the log trace: deviations which are justified by a responsibility are considered correct behaviours rather than errors.

In this paper we define a formal framework accounting for responsibilities in a process model and defining cost functions accounting for them to evaluate process alignments.

Keywords

Process Alignment, Responsibilities, Responsibility Alignment

1. Introduction

When it comes to process model executions, it might be the case that the actual executions do not perfectly match with a run foreseen by the model. Conformance checking in process mining aims at finding the process run which is closest to an execution [1]. Specifically, in classical trace alignment the approach is to scan stepwise a logged trace with a sequence of activities from the process model trying to match them and, where this is not possible, detecting mismatches.

In performing this task, most of the approaches consider control flow information only, and attribute the same cost to the deviations. As a result, the optimal alignment is the one minimizing the number of deviations. More recent works consider contextual information to evaluate mismatches [2, 3, 4, 5, 6, 7]. The underlying idea is that when additional information is available, leveraging on it in conformance checking leads to more realistic and informative alignments.

22nd International Conference of the Italian Association for Artificial Intelligence (AIxIA 2023) - Discussion Papers

*Corresponding author.

matteo.baldoni@unito.it (M. Baldoni); cristina.baroglio@unito.it (C. Baroglio); elisa.marengo@unito.it (E. Marengo); roberto.micalizio@unito.it (R. Micalizio)

0000-0002-9294-0408 (M. Baldoni); 0000-0002-2070-0616 (C. Baroglio); 0000-0003-1879-2088 (E. Marengo); 0000-0001-9336-0651 (R. Micalizio)

In this paper we focus on responsibilities, through which organizations are capable of incorporating regulations, laws, policies, objectives and such like. Responsibilities, which can be on simple tasks as well as complex tasks (i.e., subprocesses), are an integral part of business organizations. Several models exists to define them, such as Business Motivation Models (BMM) [8], assigning responsibilities to Organizational Units, or RACI matrices, associating roles to tasks with the meaning of defining which roles are directly responsible for the completion of what tasks. Responsibilities are therefore capable of capturing when an activity should, or should not, be performed. However, despite their informative power, so far conformance checking techniques, and trace alignment in particular, have neglected them. Consider, for instance, a buying and selling process. An employee is not only responsible for sending a payment confirmation after a payment, but she is also responsible for sending the confirmation only if the payment occurred. Therefore, if in an actual execution both payment and payment confirmation are missing, only the first should be considered as an anomaly. These considerations are not part of standard approaches, which instead would consider both mismatches as non-compliant behaviors. This is also due to the nature of process models, which is often more prescriptive than strictly required.

In this work, we propose to complement such a model with responsibilities ans we show how this would allow us to determine when a mismatch in an alignment is not necessarily a misbehavior, but an acceptable alternative. The paper is organized as follows. In Section 2 we propose a formalism for responsibility representation which supports the specification of declarative orderings among activities. In Section 3 we provide the formalization of process model extended with responsibilities. An alignment strategy that accounts for mismatches with the process model as well as responsibilities that are either satisfied or neglected is presented in Section 4. An extended version of the paper can be found in [9].

2. Responsibilities: Definition and Evaluation in a Trace

The term responsibility is associated with multiple shades of meaning [10]. In this paper, as well as in BMM and other business models, responsibility refers to an actor's duty to perform a task in a given context, or role responsibility in the terminology by Vincent [10]. Formally, we represent a responsibility relation as $\mathrm{R}(x,\mathsf{u},\mathsf{v})$ where x is a role, u is a context condition, and v is the duty assigned to x. Intuitively, $\mathrm{R}(x,\mathsf{u},\mathsf{v})$ states that any actor playing role x will be receptive to the need of bringing about v if u holds, and hence it will be answerable about v in that specific context. That is, it would be possible to ask x an account about her involvement in the achievement, or not, of duty v. Condition u and duty v can both be simple activities or temporal patterns on activities executions.

Our proposal is to leverage on responsibilities, and the fact that they can be neglected, as a preference criterion on alignments. In fact, in the real execution of a process, an actor can either meet or neglect her responsibilities. Our goal is to use these events as contextual information for trace alignment, relying on the assumption that role players will act so to be aligned as much as possible with their responsibilities. Let us consider an example inspired by [11].

Example (Alignments and Responsibilities). In a Fine Management Process, a fine is first sent (Send-Fine) to the offender, then the offender can either appeal to the judge (Appeal-Judge), or pay (Pay), in which case a receipt is produced by an employee (Send-Receipt). Only two model runs are possible: E1 = ⟨Send-Fine, Appeal-Judge⟩ and E2 = ⟨Send-Fine, Pay, Send-Receipt⟩. Let us consider the observed execution trace: T = ⟨Send-Fine⟩. The possible alignments with trace T are the following, where $\gg$ represents mismatches (i.e., moves where either the log trace or the model moves one step).

$$\mathrm{A1} = \begin{array}{c|c} \text{Send-Fine} & \gg \\ \hline \text{Send-Fine} & \text{Appeal-Judge} \end{array} \qquad \mathrm{A2} = \begin{array}{c|c|c} \text{Send-Fine} & \gg & \gg \\ \hline \text{Send-Fine} & \text{Pay} & \text{Send-Receipt} \end{array}$$

Classical approaches would conclude that the model execution E1 is closer to the trace because in A1 there is one mismatch, while in A2 there are two. Let us now assume that the model is complemented with an explicit representation of responsibilities, and that the employee is (always) responsible for producing a receipt only after the payment of the fine, and only in case it occurs. Assessing the two alignments against such a responsibility allows us to observe that the lack of Send-Receipt in A2 is actually correct, justified by the fact that the payment did not occur. Therefore, it should not be considered a mismatch, and the two alignments can be considered as equivalent in terms of number of mismatches.

Responsibilities provide, in a declarative manner, the expected context of an activity, which is precious for interpreting a logged trace in a way that goes beyond the syntactic distance between strings. Accordingly, we define the cost of an alignment as depending both on the responsibilities that are neglected, and the found mismatches as follows: a neglected responsibility amounts to a cost accumulated by the alignment; a mismatch justified by responsibilities (as for Send-Receipt in our example) does not contribute to the cost of an alignment, while other mismatches are considered as misbehavior and contribute to the cost.

To express context conditions and duties in a responsibility relation we rely on precedence logic defined in [12] and summarized in the following.

Precedence Logic. Given a responsibility R(x,u,v), we denote the conditions u and v as precedence logic expressions, defined over the set of symbols $\Sigma \cup \{0, \top\}$; here, Σ is a set of activity symbols, 0 means false, and $\top$ means true. Precedence logic is an event-based linear temporal logic, obtained from propositional logic augmented with the temporal operator $(\cdot)$ before. Such an operator is used to express minimal ordering requirements between events. For instance, $a \cdot b$ expresses the requirement for event a to occur some time before the occurrence of event b (need not be strictly before). Besides the before operator, the logic includes the $\vee$ (choice) and the $\wedge$ (interleaving) operators (capturing that two conditions need to be satisfied but there is no temporal requirements between them). Given a workflow u expressed in precedence logic, the residual of u against an event e, denoted as u/e, defines the evolution of u after the occurrence of event e. The residual operator is defined by rules (a – h) below, defined in [13, 12]. Here, u is a given workflow, e is an event or $\top$, its complement $\bar{e}$ represents the non-occurrence of e, and Γ_u represents the set of literals in u and their complements (e.g., $\Gamma_{a \cdot b} = \{a, \bar{a}, b, \bar{b}\}$).

The residual u/e is defined as:

(a)	$0/e \doteq 0$	(b)	$\top/e \doteq \top$
(c)	$(u_1 \wedge u_2)/e \doteq ((u_1/e) \wedge (u_2/e))$	(d)	$(u_1 \vee u_2)/e \doteq ((u_1/e) \vee (u_2/e))$
(e)	$(e \cdot u_1)/e \doteq u_1$ if $e \notin \Gamma_{u_1}$	(f)	$(u_1/e) \doteq u_1$ if $e \notin \Gamma_{u_1}$
(g)	$(e' \cdot u_1)/e \doteq 0$ if $e \in \Gamma_{u_1}$	(h)	$(\bar{e} \cdot u_1)/e \doteq 0$

Since 0 amounts to false, and $\top$ to true, the residual operator can be used for assessing whether a workflow expression u is satisfied by a given sequence of events $\sigma = \langle \sigma_1, \ldots, \sigma_m \rangle$ in Σ. Specifically, we denote as u/σ the expression $(((u/\sigma_1)/\sigma_2) \ldots)/\sigma_m$. When u/σ leads to $\top$, σ is a possible execution run of u. When u/σ leads to 0 σ represents a trace not compliant with u. According to [12], it is assumed that i) the events in σ are non-repeating (timestamps can be used to differentiate multiple instances of the same event [12]), and ii) an event e and its complement $\bar{e}$ are mutually exclusive in every sequence σ.

Evaluate Responsibilities in a Trace. Relying on precedence logic gives us two advantages: 1) generality, since we can model both contexts and duties as workflows, and 2) semantics, since we can assess the state of a responsibility against a log trace relying on the residual operator. Specifically, we can assess the state of R(x,u,v) as either i) active, ii) discharged iii) neglected, or iv) satisfied, given an execution trace $\sigma = \langle \sigma_1, \ldots, \sigma_m \rangle$ of events over Σ. Formally, let us denote as $\langle \sigma_1, \ldots, \sigma_i \rangle$ a prefix of σ events with $1 \leq i \leq m$,

- R(x,u,v) is active at step i (s.t. $i < m$), if $\mathsf{u}/\langle \sigma_1, \ldots, \sigma_i \rangle = \top$ and $\mathsf{v}/\langle \sigma_1, \ldots, \sigma_i \rangle$ is neither $\top$ nor 0;
- R(x,u,v) is discharged at step i if $\mathsf{u}/\langle \sigma_1, \ldots, \sigma_i \rangle = 0$ (the residual of v is irrelevant);
- R(x,u,v) is satisfied at step i if $\mathsf{u}/\langle \sigma_1, \ldots, \sigma_i \rangle = \top$ and $\mathsf{v}/\langle \sigma_1, \ldots, \sigma_i \rangle = \top$;
- R(x,u,v) is neglected at step i if $\mathsf{u}/\langle \sigma_1, \ldots, \sigma_i \rangle = \top$ and $\mathsf{v}/\langle \sigma_1, \ldots, \sigma_i \rangle = 0$, or at step m (the end of the execution) when $\mathsf{u}/\sigma = \top$ and v/σ is not $\top$.

Intuitively, when the responsibility is active there is an expectation on x to bring about v since the context condition u holds. When the responsibility is discharged, instead, the context condition does not hold (and cannot hold along the given σ), and hence no expectation about v can be made. The responsibility is satisfied along σ when both u and v progress to $\top$. Finally, a responsibility is neglected either when, at any execution step, the context condition u holds and the duty v does not, or when, at the end of the trace, u holds and v has not progressed to $\top$, that is, the expectation created with u has not been met.

Example (Responsibilities). Let us consider a set Σ of activity symbols $\{\mathsf{p}, \mathsf{sf}, \mathsf{sr}\}$ standing respectively for Pay, Send-Fine and Send-Receipt. Consider a responsibility relation R(x,$\top$, $\mathsf{p} \cdot \mathsf{sr}$) expressing that the receipt has to be sent only after the payment. Let us consider the execution $\langle \mathsf{sf}, \mathsf{sr} \rangle$ and apply the residual with respect to it. First, since $\mathsf{sf} \notin \Gamma_{\mathsf{p} \cdot \mathsf{sr}}$, rule (e) applies: $\mathsf{p} \cdot \mathsf{sr}/\mathsf{sf} = \mathsf{p} \cdot \mathsf{sr}$. Then, rule (g) applies to $\mathsf{p} \cdot \mathsf{sr}/\mathsf{sr}$ since $\mathsf{sr} \in \Gamma_{\mathsf{sr}}$, bringing the responsibility to be neglected.

3. Process Model with Responsibilities

In our approach a process model accounts both for the control flow, and for responsibility relations assigned to roles taking part to the process. We distinguish the two parts, defining a Process Net, specified as a labeled Petri Net in Definition 3.1; and complementing it with a set of responsibilities annotating it.

We define a process net as an extension of the process model given in [3] by including a set of roles and assigning them to the activities. A role can be seen as a participant to the process and defined in terms of its function or skills.

Definition 3.1 (Process Net). A Process Net is a Labeled Petri Net defined as a tuple $\mathsf{N} = \langle P, T, F, m_0, m_f, \Sigma, \lambda, Z, \zeta \rangle$, where P is the set of places, T is the set of transitions (with $P \cap T = \emptyset$), F is the flow relation $F \subseteq (P \times T) \cup (T \times P)$, m_0 is the initial marking, m_f is the final marking, Σ is the set of activity symbols, $\lambda : T \to \Sigma \cup \{\tau\}$ labels every transition by an activity or as silent, Z is the set of roles, and $\zeta : \Sigma \to Z$ assigns a role to every activity in Σ.

A process net N sets the scope of responsibility relations, since it specifies both the roles Z and the activities Σ over which a responsibility is defined. Responsibilities are defined at design time, and relying on precedence logic allows us to specify that a responsibility be active when a precise execution path occurs. For instance, given a process net N and the activities $\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{d} \in \Sigma$, and a role $x \in Z$, to specify that an actor playing role x is responsible for activity d only if activities a, b, and c (in the order but possibly interleaved with other events) occur, one can specify the responsibility relation $\mathrm{R}(x,\ \mathsf{a} \cdot \mathsf{b} \cdot \mathsf{c},\ \ \mathsf{d})$. Instead, to specify that the responsibility is activated when the three activities occur in any order one can use the relation $\mathrm{R}(x,\ \mathsf{a} \wedge \mathsf{b} \wedge \mathsf{c},\ \mathsf{d})$.

We expect that each responsibility is consistently defined with the process model it refers to. In other terms, both the context and the duty conditions are assumed to be (sub)workflows that can be generated by at least one model run. Therefore, there is always at least a way to satisfy a responsibility.

Activities are part of the context in which responsibilities hold. For instance, by accepting an order, an employee becomes responsible for a number of duties. We "attach" responsibilities to activities, meaning intuitively that a responsibility gets relevant when its corresponding activity is performed. For instance, $\mathrm{R}(x,\ \mathsf{a} \cdot \mathsf{b} \cdot \mathsf{c},\ \ \mathsf{d})$ can be attached to e, to express that it gets relevant when e is executed. The context condition captures that, if e can be reached from more than one path, the responsibility is activated only by the path where $\mathsf{a} \cdot \mathsf{b} \cdot \mathsf{c}$ holds. Definition 3.2 formally define the responsibility labelling of a process net.

Definition 3.2 (Responsibility Labelling). Let N be a process net, and let Z and Σ be, respectively, the set of roles and activity symbols in N. A responsibility labeling over N is a function $\mathsf{R} : \Sigma \to \{\mathsf{R}_1, \dots, \mathsf{R}_n\}$ where each R_i is a responsibility relation $\mathrm{R}(x_i, \mathsf{u}_i, \mathsf{v}_i)$, such that: $x_i \in Z$ and u_i and v_i are precedence logic expressions over $\Sigma \cup \{0, \top\}$.

A process model is then defined as a pair $\mathsf{M} = \langle \mathsf{N}, \mathsf{R} \rangle$ where N is a process net as in Definition 3.1, and R is a responsibility labelling as in Definition 3.2.

4. Flow and Responsibility Alignments

An alignment compares a process execution against an execution trace (i.e., a log trace). Generally, the objective is to find, among the possible ones, an alignment which is optimal w.r.t. a criterion of preference. Intuitively, an alignment proceeds step-by-step on the model and on the log: at each step, if the activity in the model and the one in the log match each other, a synchronous move is made, and both model and log advance one step. Otherwise, either the model moves and the log does not, or the other way around, the log moves and the model does not. Usually, to find an optimal matching, a cost function associated with mismatches (i.e., asynchronous moves) is defined. So, an optimal alignment is the one minimizing the cumulative cost of the mismatches. The classical approach is to minimize the number of asynchronous moves [1].

In our approach, an optimal alignment is determined taking into account both the alignment between a log trace and a model run, and the involved responsibilities. We refer to the former as flow alignment and to the latter as responsibility alignment. Definition 4.1 formalizes the notion of flow alignment. The symbol $\gg$ represents a no-move, and is used for marking asynchronous moves. More in general, given a process model $\mathsf{M} = \langle \mathsf{N}, \mathsf{R} \rangle$, we use the term model run for the sequence of activity symbols in Σ produced by a full run of the process net N, where a Petri Net full run is a sequence of firings from the initial marking to the final one [3]. We also assume the process net N to be easy sound [1], that is, there exists at least one full run. The term log trace, instead, refers to an actual execution of a process instance of M. It is a finite sequence of activity symbols $\sigma \in \Sigma^*$ (i.e., the space of sequences defined over Σ).

Definition 4.1 (Flow Alignment). Let $\sigma = \langle \sigma_1, \dots, \sigma_m \rangle$ be a log trace in Σ^*, and $\mathsf{N} = \langle P, T, F, m_0, m_f, \Sigma, \lambda, Z, \zeta \rangle$ a process net, an alignment of σ with the process net N is a finite sequence $\varphi = \langle (\sigma'_1, u'_1), \dots, (\sigma'_p, u'_p) \rangle$ of moves such that:

- each move is either: a synchronous move $(a,t) \in \Sigma \times T$ with $a = \lambda(t)$, a log move $(a, \gg)$, or a model move $(\gg, t)$,
- dropping the $\gg$ symbols from the left projection $(\sigma'_1, \dots, \sigma'_p)$ of φ, yields σ,
- dropping the $\gg$ symbols from the right projection $(u'_1, \dots, u'_p)$ of φ, yields a full run u of N.

To consider the responsibilities in evaluating the optimal alignment, our approach is to collect all the responsibilities attached to the activities of a model run (i.e., the responsibilities that should be satisfied along a possible, expected execution), and assess them against a log trace (to check if indeed they are satisfied). The cost of an alignment, thus, takes also into account the cost of neglected responsibilities. Moreover, the responsibilities collected along a model run give us a context for assessing whether a model move (i.e., a "skip" on the log side) actually represents an execution error, or a proper behavior.

Given a flow alignment φ, its responsibility set is the set of responsibilities attached to the activities in the model run given by the right projection of φ (i.e., the model-side projection). In general, a responsibility set can be computed for any non-empty prefix of φ by considering the alignment up to a given step j.

Definition 4.2 (Responsibility Set). Let $\varphi = \langle(\sigma'_1, u'_1), \dots, (\sigma'_p, u'_p)\rangle$ be a flow alignment between a process model $\mathsf{M} = \langle\mathsf{N}, \mathsf{R}\rangle$ and a log trace $\sigma \in \Sigma^*$, the responsibility set $\mathscr{R}^{\varphi,j}$ for the alignment φ at step j ($1 \leq j \leq p$) is defined as $\mathscr{R}^{\varphi,j} = \cup_{i=1}^{j} \mathsf{R}(\lambda(u'_i))$.

It holds $\mathsf{R}(\gg) = \emptyset$. As a shortcut, we denote as $\mathscr{R}^{\varphi}$ the set $\mathscr{R}^{\varphi,p}$, that is the set of responsibilities computed considering all the steps in the alignment φ. The responsibilities in $\mathscr{R}^{\varphi}$ are actually satisfied or neglected depending on the activities that are included in the log trace (i.e., log-side projection of φ). Thus, we first extend the notion of residuation of the precedence logic to responsibility relations, and then to a responsibility set.

Given a responsibility set $\mathscr{R}^{\varphi} = \{\mathsf{R}_1, \dots, \mathsf{R}_k\}$ with $\mathsf{R}_i = \mathrm{R}(\mathrm{x}_i, \mathsf{u}_i, \mathsf{v}_i)$, let $\sigma' = \langle\sigma'_1, \dots, \sigma'_p\rangle$ be the log-side projection of φ. Then, the notation R_i/σ' is a shorthand for $\mathrm{R}(\mathrm{x}_i, \mathsf{u}_i/\sigma', \mathsf{v}_i/\sigma')$ and $\mathscr{R}^{\varphi}/\sigma'$ is a shorthand for $\{\mathsf{R}_1/\sigma', \dots, \mathsf{R}_k/\sigma'\}$. Additionally, the residuation of any expression u with $\gg$ has no effect on the expression, namely $\mathsf{u}/\gg = \mathsf{u}$.

It is worth noting that the rewriting rules of the precedence logic guarantee a consistent evaluation of the responsibilities against a log trace. In fact, whenever a responsibility progresses from the active state to either satisfied, neglected, or discharged, such a second state is final: the state of the responsibility can no longer evolve along the same trace. That is, further events along the trace cannot satisfy a neglected responsibility nor vice versa. At the same time, a responsibility activated along a log trace must necessarily evolve to either satisfied or neglected by the end of the same trace (see [9] for details).

Example (Responsibility Set). Consider the example in Section 2, the responsibility set $\mathscr{R}^{A1}$ of alignment $\mathscr{R}^{A2}$ for A2 is given by the responsibilities associated to Send-Fine, Pay and Send-Receipt. The resulting sets will be residuated with respect to the log trace (i.e., activity Send-Fine).

4.1. Cost functions for optimal alignments.

In general, several alignments of a log trace w.r.t. a model exist. To compare them and determine the optimal one we define a cost function that considers both the cost of the mismatches between the model run and the log trace, which we call Flow Alignment Cost $\mathscr{C}_{\mathsf{N},}$, and the cost for the neglected responsibilities, which we call Responsibility Alignment Cost $\mathscr{C}_{\mathrm{R},}$.

The cost $\mathscr{C}_{\mathrm{R},}$ corresponds to the number of responsibilities that are neglected in a flow alignment φ. To compute them, first the responsibility set $\mathscr{R}^{\varphi}$ for φ is determined (Definition 4.2). Then, $\mathscr{R}^{\varphi}$ is residuated with respect to the projection log-side of φ. Neglected responsibilities are then those that are active, but not satisfied at the end of the trace.

Example (Responsibility Alignment Cost). Let us consider the fine process scenario, and assume that the employee is responsible for archiving (ar) any sent fine (sf) after 60 days. Now, in both alignments A1 and A2 (see Section 2), the responsibility is activated but is not satisfied by the end of both alignments. Thus is marked as neglected and brings a cost in both alignments.

The flow cost $\mathscr{C}_{\mathsf{N},}$, calculates the cost of every mismatch (i.e., either model or log moves) included within a given alignment. Notably, this calculation takes into account responsibilities as a sort of context. By using them, in fact, we are able to identify some model moves as correct, and not as mismatches. Intuitively, to compute $\mathscr{C}_{\mathsf{N},}$, each alignment step (σ'_j, u'_j) of a flow alignment φ is considered. A step is a mismatch, and hence has to be counted as a cost, when it is either a log move or it is a model move which is not justified by any responsibility. A model move is justified by a responsibility if executing that activity in the log (instead of a $\gg$) would have lead an active responsibility to progress to neglected. This means that skipping the activity is consistent with at least one responsibility, and hence does not represent a misbehavior.

Example (Flow Alignment Cost). Let us consider again alignment A2 in Section 2, and let us assume that $\mathrm{R}(\mathrm{x},\top,\ \mathsf{p}\cdot\mathsf{sr})$ is associated with activity Pay to indicate that, if the expected execution is the one that goes through Pay rather than Appeal-Judge, then the receipt has to be sent only after and only in case of a payment. Concerning the Flow Alignment Cost, albeit A2 has two asynchronous model moves, only the first one actually contributes to the flow cost. The log skip on Send-Receipt (sr), instead, is justified since the occurrence of sr in the log would lead to the violation of $\mathrm{R}(\mathrm{x},\top,\ \mathsf{p}\cdot\mathsf{sr})$.

The total cost of an alignment φ is computed as the weighted sum of the flow and the responsibility costs: $\mathscr{C}_{\varphi} = \gamma \cdot \mathscr{C}_{\mathsf{N},} + \delta \cdot \mathscr{C}_{\mathrm{R},}$

Coefficients γ and δ are domain-dependent weights that can be tuned for penalizing more either neglected responsibilities or asynchronous moves. An alignment between a model M and a log trace σ is optimal if $\mathscr{C}_{\varphi}$ is minimal.

5. Conclusions and Future Work

We presented a novel methodology of process alignment which takes into account responsibilities during the search for optimal alignments. Several proposals in the literature focus on extending process alignment considering other perspectives besides the control flow [3, 2, 14, 4, 15, 5, 6, 11]. To the best of our knowledge, no approach considers the perspective of responsibilities as a perspective or for evaluating model moves.

An explicit representation of responsibilities opens several future directions. First, each alignment found by our algorithm is associated with a set of met and unmet responsibilities. These two sets provide a sort of justification why a specific alignment has been selected as optimal. Moreover, by considering the set of unmet responsibilities one could reason about possible inefficiencies and flaws, enabling a responsibility-driven procedure for re-engineering a process. In addition, role responsibilities designate the actors playing a specific role as “in charge” of some job, and hence capable of providing accounts about its accomplishment or failure. As future work, our responsibility framework can be complemented with accountability relationships [16] to improve both the computation of alignments and their understanding in the context of a business organization.

Acknowledgments. This work is part of the projects NODES, which has received funding from the MUR-M4C2 1.5 of PNRR funded by the European Union-NextGenerationEU (Grant agreement no. ECS00000036) and “APE : Accountability and Processes from Events”, financed by MUR “Grant for Internationalization” call.

References

[1] J. Carmona, B. F. van Dongen, A. Solti, M. Weidlich, Conformance Checking - Relating Processes and Models, Springer, 2018.

[2] G. Acitelli, M. Angelini, S. Bonomi, F. M. Maggi, A. Marrella, A. Palma, Context-Aware Trace Alignment with Automated Planning, in: ICPM 2022, 2022, pp. 104–111.

[3] M. Boltenhagen, T. Chatain, J. Carmona, A Discounted Cost Function for Fast Alignments of Business Processes, in: BPM, volume 12875 of LNCS, 2021, pp. 252–269.

[4] M. Alizadeh, X. Lu, D. Fahland, N. Zannone, W. M. P. van der Aalst, Linking data and process perspectives for conformance analysis, Comput. Secur. 73 (2018).

[5] A. S. Mozafari Mehr, R. M. de Carvalho, B. F. van Dongen, Detecting Complex Anomalous Behaviors in Business Processes: A Multi-perspective Conformance Checking Approach, in: ICPM Workshops, LNBIP 468, Springer, 2022, pp. 44–56.

[6] G. Di Federico, A. Burattin, Do You Behave Always the Same? - A Process Mining Approach, in: ICPM Workshops, LNBIP 468, Springer, 2022, pp. 5–17.

[7] M. Baldoni, C. Baroglio, F. Capuzzimati, E. Marengo, V. Patti, A Generalized Commitment Machine for 2CL Protocols and its Implementation., 7784 LNAI (2013) 96–115.

[8] OMG, BMM Model, 2015. URL: https://www.omg.org/spec/BMM/1.3/PDF.

[9] M. Baldoni, C. Baroglio, E. Marengo, R. Micalizio, A Responsibility Framework for Computing Optimal Process Alignments, in: 7th International Workshop in Artificial Intelligence for Business Process Management, Springer, 2023.

[10] N. Vincent, A structured taxonomy of responsibility concepts, Moral responsibility: Beyond free will and determinism (2010).

[11] F. Mannhardt, M. de Leoni, H. A. Reijers, W. M. P. van der Aalst, Balanced Multi-Perspective Checking of Process Conformance, Computing 98 (2016).

[12] M. P. Singh, M. N. Huhns, Service-Oriented Computing - Semantics, Processes, Agents, Wiley, 2005.

[13] M. Baldoni, C. Baroglio, R. Micalizio, S. Tedeschi, Accountability in Multi-Agent Organizations: From Conceptual Design to Agent Programming, Auton. Agents Multi Agent Syst. 37 (2023) 7.

[14] V. Bloemen, S. J. van Zelst, W. M. P. van der Aalst, B. F. van Dongen, J. van de Pol, Aligning Observed and Modelled Behaviour by Maximizing Synchronous Moves and Using Milestones, Inf. Syst. 103 (2022) 101456.

[15] P. Felli, A. Gianola, M. Montali, A. Rivkin, S. Winkler, CoCoMoT: Conformance Checking of Multi-perspective Processes via SMT, in: BPM 2021, volume 12875 of LNCS, Springer, 2021, pp. 217–234.

[16] M. Baldoni, C. Baroglio, R. Micalizio, S. Tedeschi, Robustness based on accountability in multiagent organizations, in: AAMAS, IFAAMAS, 2021, pp. 142–150.