A Responsibility Framework for Computing Optimal Process Alignments

Matteo Baldoni[0000−0002−9294−0408], Cristina Baroglio[0000−0002−2070−0616], Elisa Marengo[0000−0003−1879−2088], and Roberto Micalizio[0000−0001−9336−0651]

Computer Science Department – University of Turin, Turin, Italy
`firstname.lastname@unito.it`

Abstract. In this paper we propose a novel approach to process alignment which leverages on contextual information captured by way of *responsibilities*. On the one hand, responsibilities may justify deviations. In these cases, we consider deviations as correct behaviors rather than errors. On the other hand, responsibilities can either be met or neglected in the execution trace. Thus, we prefer alignments where neglected responsibilities are minimized.
The paper proposes a formal framework for responsibilities in a process model, including the definition of cost functions to determine optimal alignments. It also outlines a branch-and-bound algorithm for their computation.

Keywords: Process Alignment · Responsibilities · Responsibility Alignment

1 Introduction

Trace alignment is a fundamental activity of conformance checking in process mining [11]. It aims at relating an intended behavior, described by way of a process model, and an actual execution trace recorded in an event log. Trace alignment highlights where the actual trace deviates from the process model, and provides insights for further investigations such as performance analysis [1], model repair [11], diagnosis and such like.

Many consolidated approaches (see [11] for an overview) focus on trace alignment from the *control flow perspective*. These approaches try to match a logged trace with a sequence of activities from a process model by scanning both stepwise, looking for mismatches. In the classical approaches, all mismatches have the same weight, and an optimal alignment is one that minimizes them. More recent works consider additional aspects besides the control flow, such as contextual information to weight the impact of a mismatch on the alignment considering when the mismatch occurs [2,9]. Other works show the importance of considering additional perspectives besides the control flow, such as data access [4,15] and temporal aspects on the occurrence of the events [12,5]. In general, the idea is that when additional information is available, leveraging on it in conformance checking leads to more realistic and informative alignments.

In this paper we focus on *responsibilities*, through which organizations gain the flexibility that a prescriptive representation of a process cannot enjoy. Via responsibilities, in fact, organizations are capable of incorporating (internal or external) regulations, laws, policies, refine objectives and such like.

Responsibilities are captured in different ways within a business organization. RACI matrices, for instance, specify which roles are directly responsible for the completion of what tasks. In the *Business Motivation Model* (BMM) by OMG [16] responsibilities are assigned to *Organizational Unit.* Note that this applies not only to simple tasks, but also every *Business Process* (or complex task) must be under the responsibility of some Organizational Unit.

Responsibilities, thus, are an integral part of business organizations, but so far conformance checking techniques, and trace alignment in particular, have neglected their informative power. Responsibilities, in fact, give us useful insights on when an activity should, or should not, be performed. Consider, for instance, a buying and selling process. An employee is not only responsible for sending a payment confirmation after a payment, but (s)he 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. Standard approaches considering the activities mismatches would instead count both as non-compliant behaviors. This is also due to the nature of process models, which is often more prescriptive than strictly required. Complementing such a model with responsibilities would allow us to understand that a mismatch in an alignment is not necessarily a misbehavior in the process execution, but an acceptable alternative.

In this paper we propose: *i)* a formalism for responsibility representation which supports the specification of declarative orderings among activities; *ii)* an alignment strategy that accounts for mismatches with the process model as well as responsibilities that are either satisfied or neglected; *iii)* we outline an algorithm to compute all the optimal alignments.

In Section 2 we introduce the responsibility relations and their formalization. Section 3 formalizes the concept of process model extended with the responsibilities. Process and responsibilities alignments are presented in Section 4. Related work in Section 5 and Conclusions in Section 6 end the paper.

2 Responsibilities: Definition and Evaluation in a Trace

The term responsibility is associated with multiple shades of meaning [18]. 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 [18]. Formally, we represent a responsibility relation as $R(x,\mathsf{u},\mathsf{v})$ where x is a role, u is a context condition, and v is the duty assigned to x. Intuitively, $R(x,\mathsf{u},\mathsf{v})$ 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 [14].

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: $E1 = \langle \textit{Send-Fine}, \textit{Appeal-Judge} \rangle$ and $E2 = \langle \textit{Send-Fine}, \textit{Pay}, \textit{Send-Receipt} \rangle$. Let us consider the observed execution trace: $T = \langle \textit{Send-Fine} \rangle$. The possible alignments with trace T are the following, where $\gg$ represents mismatches (i.e., moves where either the log trace or the model move one step).

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

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 [17] and summarized in the following.

Precedence Logic. Given a responsibility $R(\mathsf{x},\mathsf{u},\mathsf{v})$, we denote the conditions u and v as precedence logic expressions [17], 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 [7,17]. 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 [17], it is assumed that *i)* the events in σ are non-repeating (timestamps can be used to differentiate multiple instances of the same event [17]), 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 $R(\mathsf{x},\mathsf{u},\mathsf{v})$ 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 $1 \leq i \leq m$,

- $R(\mathsf{x},\mathsf{u},\mathsf{v})$ 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;
- $R(\mathsf{x},\mathsf{u},\mathsf{v})$ is *discharged* at step i if $\mathsf{u}/\langle \sigma_1, \ldots, \sigma_i \rangle = 0$ (the residual of v is irrelevant);
- $R(\mathsf{x},\mathsf{u},\mathsf{v})$ is *satisfied* at step i if $\mathsf{u}/\langle \sigma_1, \ldots, \sigma_i \rangle = \top$ and $\mathsf{v}/\langle \sigma_1, \ldots, \sigma_i \rangle = \top$;
- $R(\mathsf{x},\mathsf{u},\mathsf{v})$ 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 $R(\text{x}, \top, \ \mathsf{p} \cdot \mathsf{sr})$ 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 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 [9] 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 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 $R(x, \ \mathsf{a} \cdot \mathsf{b} \cdot \mathsf{c}, \ \mathsf{d})$. Instead, to specify that the responsibility is activated when the three activities occur in any order one can use the relation $R(x, \ \mathsf{a} \wedge \mathsf{b} \wedge \mathsf{c}, \ \mathsf{d})$.

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, $R(x, \ \mathsf{a} \cdot \mathsf{b} \cdot \mathsf{c}, \ \mathsf{d})$ 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 2 formally define the responsibility labelling of a process net.

Definition 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 $R : \Sigma \rightarrow \{\mathsf{R}_1, \ldots, \mathsf{R}_n\}$ where each R_i is a responsibility relation $R(x_i, \mathsf{u}_i, \mathsf{v}_i)$, 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 1, and R is a responsibility labelling as in Definition 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 [11].

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 3, adapted from [9], formally introduces the notion of *flow alignment*, capturing the alignment between a process net and an execution trace. 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 [9]. We also assume the process net N to be *easy sound* [11], 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 therefore a finite sequence of activity symbols $\sigma \in \Sigma^*$ (i.e., the space of sequences defined over symbols in Σ).

Definition 3 (Flow Alignment). *Let $\sigma = \langle \sigma_1, \ldots, \sigma_m \rangle$ 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 $\varphi = \langle (\sigma'_1, u'_1), \ldots, (\sigma'_p, u'_p) \rangle$ 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 $(\sigma'_1, \ldots, \sigma'_p)$ of φ, yields σ,*
– *dropping the $\gg$ symbols from the right projection $(u'_1, \ldots, u'_p)$ 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 (Responsibility Set). *Let $\varphi = \langle(\sigma'_1, u'_1), \ldots, (\sigma'_p, u'_p)\rangle$ 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 $\mathscr{R}^{\varphi} = \{\mathsf{R}_1, \ldots, \mathsf{R}_k\}$ with $\mathsf{R}_i = R(\mathrm{x}_i, \mathsf{u}_i, \mathsf{v}_i)$, let $\sigma' = \langle \sigma'_1, \ldots, \sigma'_p\rangle$ be the log-side projection of φ. Then, the notation R_i/σ' is a shorthand for $R(\mathrm{x}_i, \mathsf{u}_i/\sigma', \mathsf{v}_i/\sigma')$ and $\mathscr{R}^{\varphi}/\sigma'$ is a shorthand for $\{\mathsf{R}_1/\sigma', \ldots, \mathsf{R}_k/\sigma'\}$. Additionally, the residuation of any expression u with $\gg$ has no effect on the expression, namely $\mathsf{u}/\gg = \mathsf{u}$.

Proposition 1. *[Consistency] Let $\mathscr{R}^{\varphi,j}$ be the responsibility set computed at step j $(1 \leq j \leq p)$ of an alignment $\varphi = \langle(\sigma'_1, u'_1), \ldots, (\sigma'_p, u'_p)\rangle$, let σ' be the projection log-side of φ, then for each $\mathsf{R}_i \in \mathscr{R}^{\varphi,j}$ the following conditions hold:*

1. *if $\mathsf{R}_i/\langle \sigma'_1, \ldots, \sigma'_j\rangle$ is satisfied, then also R_i/σ' is satisfied;*
2. *if $\mathsf{R}_i/\langle \sigma'_1, \ldots, \sigma'_j\rangle$ is neglected, then also R_i/σ' is neglected;*
3. *if $\mathsf{R}_i/\langle \sigma'_1, \ldots, \sigma'_j\rangle$ is discharged, then also R_i/σ' is discharged;*
4. *if $\mathsf{R}_i/\langle \sigma'_1, \ldots, \sigma'_j\rangle$ is active, then R_i/σ' is either satisfied or neglected.*

This proposition, that follows directly from the rewriting rules of the precedence logic, guarantees 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.

Example (Responsibility Set). Consider the example in Section 2, the responsibility set $\mathscr{R}^{A1}$ of alignment A1 will be the union of the responsibilities associated with the activities *Send-Fine* and *Appeal-Judge*, while the set $\mathscr{R}^{A2}$ 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*) and the cost of the alignment computed as describe in the following.

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* $\mathscr{C}_{\mathsf{N},\varphi}$, and the cost for the neglected responsibilities, which we call *Responsibility Alignment Cost* $\mathscr{C}_{R,\varphi}$.

Let us start from the latter one. The responsibility cost $\mathscr{C}_{R,\varphi}$ is computed for a flow alignment $\varphi = \langle(\sigma'_1, u'_1), \ldots, (\sigma'_p, u'_p)\rangle$ between a process model $\mathsf{M} = \langle \mathsf{N}, \mathsf{R}\rangle$ and a log trace $\sigma \in \Sigma^*$. The cost $\mathscr{C}_{R,\varphi}$ corresponds to the number of responsibilities that are neglected in φ. To compute them, first the responsibility set $\mathscr{R}^{\varphi}$ for φ is determined (Definition 4). 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. Note that the approach can be easily extended to consider different responsibility costs (for instance to capture that some responsibilities are more important to be satisfied than others).

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, independently of whether the offender has appealed to the judge or paid for the fine; that is, responsibility $R(\mathsf{x},\mathsf{sf},\mathsf{ar})$ is attached to sf. Now, in both alignments A1 and A2 (see Section 2), the responsibility is activated but, since it is not satisfied by the end of both alignments, it is marked as neglected in both cases, thus bringing a cost in both alignments.

The second component of our cost function, the flow cost $\mathscr{C}_{\mathsf{N},\varphi}$, 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. To this end, $\mathscr{C}_{\mathsf{N},\varphi}$ is computed with respect to a flow alignment $\varphi = \langle(\sigma'_1, u'_1), \ldots, (\sigma'_p, u'_p)\rangle$ between a process model $\mathsf{M} = \langle \mathsf{N}, \mathsf{R}\rangle$ and a log trace $\sigma \in \Sigma^*$. To compute $\mathscr{C}_{\mathsf{N},\varphi}$, each alignment step (σ'_j, u'_j) is considered. A step (σ'_j, u'_j) is a mismatch, and hence to be counted as a cost, when it is either a log move (i.e., the label $\lambda(u'_j)$ assigned to transition u'_j is $\gg$), or it is a model move (i.e., σ'_j equals $\gg$) which is not justified by any responsibility. A model move is justified by a responsibility if there is at least one

$R(x,\mathsf{u},\mathsf{v}) \in \mathscr{R}^{\varphi,j}$ which is not neglected at step j, but it would be if the log event σ'_j were substituted by the corresponding model activity $\lambda(u'_j)$. In other words, executing the activity corresponding to a synchronous move would have lead an active responsibility $R(x,\top,\mathsf{v})$ to progress to neglected i.e., $R(x,\top,0)$. This means that skipping activity $\lambda(u'_j)$ is consistent with at least one responsibility, and hence $\gg$ does not represent a misbehavior.

Note that this approach is compatible with other approaches in the literature where the cost of a mismatch is not necessarily constant nor one (e.g., [9,2]).

Example (Flow Alignment Cost). Let us consider again alignment A2 in Section 2, and let us assume that $R(\mathrm{x},\top,\ \mathsf{p}\cdot\mathsf{sr})$ 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 $R(\mathrm{x},\top,\ \mathsf{p}\cdot\mathsf{sr})$.

The total cost of an alignment is computed as the weighted sum of the flow and the responsibility costs.

Definition 5 (Alignment Cost, Optimal Alignment). *Let $\varphi = \langle(\sigma'_1, u'_1)$, $\ldots$, $(\sigma'_p, u'_p)\rangle$ be the flow alignment between a process model $\mathsf{M} = \langle \mathsf{N}, R\rangle$ and a log trace $\sigma \in \Sigma^*$, the cost function of the alignment φ is*

$$\mathscr{C}_\varphi = \gamma \cdot \mathscr{C}_{\mathsf{N},\varphi} + \delta \cdot \mathscr{C}_{R,\varphi}$$

An alignment between a model M and a log trace σ is optimal if $\mathscr{C}_\varphi$ is minimal.

Coefficients γ and δ are domain-dependent weights that can be tuned for penalizing more either neglected responsibilities or asynchronous moves. A greater weight for δ (responsibility coefficient) prefers asynchronous moves to neglected responsibilities. On the other hand, a greater weight for γ (flow coefficient), prefers neglected responsibilities to mismatches, thus the found alignments would be characterized by as many synchronous moves as possible even though this could lead to violate responsibilities.

4.2 Computation of the Optimal Alignment

To compute the optimal alignment, many approaches in the literature adopt A* or branch-and-bound algorithms. The approach we outline in this section exploits a branch-and-bound strategy, and relies on the Synchronous Product Net (SN) between the process net and a sequential Petri Net representing the log trace. Intuitively, the SN combines the two nets representing the synchronous and asynchronous moves. The SN is built as follows: every model (resp. log) transition is augmented with the $\gg$ symbol to represent asynchronous moves. Synchronous moves are represented with additional transitions, each labeled with the activity synchronously performed. We rely on the formal definition of SN as in [11]. In this setting, an alignment corresponds to a *full run* in the SN, that is a firing sequence bringing from the initial marking to a final marking, where the final markings are those where both the process net and the log trace reach one of their final markings.

Given in input the SN, it is possible to implement a *branch and bound* search strategy which finds the optimal full runs, i.e., those runs having minimal cost computed according to Definition 5. The idea is as follows. The search space is given by the set of possible markings of the SN; for each marking, a suitable data structure keeps *i)* the marking, *ii)* the sequence of transitions leading to it (each transition is an alignment step), *iii)* the cost of the alignment up to that point, *iv)* the set of responsibilities collected along this (partial) alignment. The algorithm keeps in a queue the frontier of the search, i.e., the markings still to be considered. The frontier is ordered in a non-decreasing way with respect to the marking cost, and it is initialized with the initial marking.

At each step of the search, the algorithm removes the top marking from the frontier (i.e., the one with the lowest cost). If the top marking is a final marking, then the cost of the responsibilities that are neglected because not satisfied at the end of the alignment (and not counted in the previous steps), need to be added to the alignment cost. Then the cost of the found solution is compared with the best cost found up to that point and, if it is equally good is added to the set of optimal solutions while if it is better it replaces the current best. Otherwise, it is discarded.

On the other hand, if the top marking is not a final marking, the algorithm computes, for each enabled transition, the reachable markings from this top marking (the top marking is now visited and will not be visited again in the future). For each reachable marking, if not yet visited, the algorithm computes the responsibility set and the cost up to this point, and then the marking is stored in the frontier in the proper order. The responsibility set is computed incrementally, by adding to the set of responsibilities collected up to that point, the responsibilities associated to the last transition. The cost associated with the marking is computed by combining flow and responsibilities costs as in Definition 5. To compute the flow cost, we count the asynchronous moves along the current alignment; in case of a model move, a cost is added only if the skip in the log is not justified by at least one responsibility associated with this marking (as explained in Section 4.1). On the other side, the responsibility cost counts all the responsibilities that, associated with this marking, are neglected by a synchronous or model move. It is worth noting that, this cost function is monotone, in fact, by Proposition 1, there is no case when the cost for a neglected responsibility needs to be reverted since the very same responsibility gets satisfied.

5 Related Work

Several proposals in the literature focus on extending process alignment considering other perspectives besides the control flow. Among these, several consider data aspects [4,13,15], time [12], costs and such like. Interestingly, in [14] the authors propose a cost function which, similarly to our proposal, combines the costs from different perspectives.

Approaches focusing on the control flow differentiate the cost of a misalignment based on when it occurs [9,2,8]. In [9] mismatches occurring at the early stages of an execution have more impact than those occurring at later stages. Authors in [2] also consider which activities occurred before and after that mismatch. The proposal in [8], instead, is to favor alignments where the number of synchronous moves is maximized. This is achieved by defining a cost function which penalizes log moves only.

Considering the technical aspects, instead, many proposals rely on Petri Nets and specifically on the Synchronous Product Net. In general, the cheapest path search techniques rely on Dijkstra- or A^*-based algorithms [9,3]; possibly, pruning techniques are applied to speed up the overall search [9,11]. Approaches based on planning [2] and SAT algorithms [10] have also been proposed.

To the best of our knowledge, no existing approaches consider the perspective of responsibilities. In our proposal, we consider it in two ways: as an additional perspective compared to the control flow and to evaluate model moves. One similarity with [8], is that we propose a strategy for not counting the cost of model moves. However, while in [8] model moves are ignored systematically, in our approach we do not consider a model move as a cost only when performing the corresponding synchronous move would lead to neglect some responsibilities.

6 Conclusions and Future Work

We presented a novel methodology of process alignment which takes into account responsibilities (see Section 2) during the search for optimal alignments. The paper contributes with a formal framework for representing responsibilities and using them for complementing a process model. A branch and bound algorithm is also discussed whose implementation is still in progress.

An explicit representation of responsibilities as part of a process model opens several future directions. First, each alignment found by our algorithm is associated with a set of met and unmet responsibilities. In first lieu, these two sets provide a sort of justification why a specific alignment has been selected as optimal. More generally, however, by considering the set of unmet responsibilities for a number of log traces, one could reason about possible inefficiencies and flaws in the process model, enabling a responsibility-driven procedure for re-engineering a process, where the responsibilities themselves could be redefined. In addition, role responsibilities designate the actors playing a specific role as "in charge" of some job, and hence capable of providing accounts about the accomplishment, or failure, of that job. This permits the creation of accountability relationships between the actor who is responsible for a job, and another actor who has some interest in that job (e.g., for her decision process) [6]. By means of accountability, problems can be detected and examined with the objective to understand why the problem has occurred. An interesting future direction, thus, is to complement our responsibility framework with accountability relationships as a way for improving 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 with grant agreement no. ECS00000036, and "APE : Accountability and Processes from Events", financed by MUR in the context of the "Grant for Internationalization" call.

References

1. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.F.: Replaying history on process models for conformance checking and performance analysis. WIREs Data Mining Knowl. Discov. **2**(2), 182–192 (2012)
2. Acitelli, G., Angelini, M., Bonomi, S., Maggi, F.M., Marrella, A., Palma, A.: Context-Aware Trace Alignment with Automated Planning. In: ICPM 2022 (2022)
3. Adriansyah, A.: Aligning Observed and Modeled Behavior. Ph.D. thesis, Technische Universiteit Eindhoven (2014)
4. Alizadeh, M., Lu, X., Fahland, D., Zannone, N., van der Aalst, W.M.P.: Linking data and process perspectives for conformance analysis. Comput. Secur. 73 (2018)
5. Baldoni, M., Baroglio, C., Capuzzimati, F., Marengo, E., Patti, V.: A Generalized Commitment Machine for 2CL Protocols and its Implementation. 7784 LNAI p. 96 – 115 (2013)
6. Baldoni, M., Baroglio, C., Micalizio, R., Tedeschi, S.: Robustness based on accountability in multiagent organizations. In: AAMAS. IFAAMAS (2021)
7. Baldoni, M., Baroglio, C., Micalizio, R., Tedeschi, S.: Accountability in Multi-Agent Organizations: From Conceptual Design to Agent Programming. Auton. Agents Multi Agent Syst. **37**(1), 7 (2023)
8. Bloemen, V., van Zelst, S.J., van der Aalst, W.M.P., van Dongen, B.F., van de Pol, J.: Aligning Observed and Modelled Behaviour by Maximizing Synchronous Moves and Using Milestones. Inf. Syst. **103**, 101456 (2022)
9. Boltenhagen, M., Chatain, T., Carmona, J.: A Discounted Cost Function for Fast Alignments of Business Processes. In: BPM. LNCS, vol. 12875, pp. 252–269 (2021)
10. Boltenhagen, M., Chatain, T., Carmona, J.: Model-Based Trace Variant Analysis of Event Logs. Inf. Syst. **102**, 101675 (2021)
11. Carmona, J., van Dongen, B.F., Solti, A., Weidlich, M.: Conformance Checking - Relating Processes and Models. Springer (2018)
12. Di Federico, G., Burattin, A.: Do You Behave Always the Same? - A Process Mining Approach. In: ICPM Workshops. pp. 5–17. LNBIP 468, Springer (2022)
13. Felli, P., Gianola, A., Montali, M., Rivkin, A., Winkler, S.: CoCoMoT: Conformance Checking of Multi-perspective Processes via SMT. In: BPM 2021
14. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.M.P.: Balanced Multi-Perspective Checking of Process Conformance. Computing **98** (2016)
15. Mozafari Mehr, A.S., de Carvalho, R.M., van Dongen, B.F.: Detecting Complex Anomalous Behaviors in Business Processes: A Multi-perspective Conformance Checking Approach. In: ICPM Workshops. pp. 44–56. LNBIP 468, Springer (2022)
16. OMG: BMM Model (2015), `https://www.omg.org/spec/BMM/1.3/PDF`
17. Singh, M.P., Huhns, M.N.: Service-Oriented Computing - Semantics, Processes, Agents. Wiley (2005)
18. Vincent, N.: A structured taxonomy of responsibility concepts. Moral responsibility: Beyond free will and determinism (08 2010)