A Technique for Collaboration Discovery

Flavio Corradini[1], Barbara Re[1], Lorenzo Rossi[1(✉)], and Francesco Tiezzi[2]

[1] School of Science and Technology, University of Camerino, Camerino, Italy
lorenzo.rossi@unicam.it

[2] Dipartimento di Statistica, Informatica, Applicazioni, University of Florence, Florence, Italy

Abstract. In the last years, researchers have contributed to the process mining domain with several techniques and tools supporting the discovery of business processes. Almost all these contributions rely on event logs stored in the information systems of single organizations. In contrast, the discovery of collaborative scenarios where the information systems are distributed among different interacting organizations has been disregarded. In this context, we propose a novel technique for discovering collaboration models from sets of event logs stored in distributed information systems. Given the distributed logs of interacting organizations, the technique discovers each organization's process through one of the available algorithms introduced by the process mining community. It also analyzes the logs to extract information on messages exchange. This information permits automatically combining the discovered processes into a collaboration diagram representing the distributed system's behavior and providing analytics on messages exchange. The technique has been implemented in a tool and evaluated via several experiments.

Keywords: BPMN collaborations · Processes discovery · Messages analysis

1 Introduction

Nowadays, organizations increasingly need to interact to achieve their goals collaboratively and create new forms of business. This requires organizations to form distributed systems, guaranteeing their interoperability. However, this task is made complex by the need to coordinate the interactions of various participants, dealing with requirements, constraints, and regulations coming from different organizations. Effective cooperation among organizations demands the compatibility of their business processes. Such cooperation can be supported by the observations of systems' behavior rather than by sharing documentation that is often incomplete and out of date [6].

In this direction, the most significant contributions come from the process mining community, referring to the automated *discovery of business process models* from data produced by IT systems, i.e., *event logs* [17]. Despite *"there is no foundational reason why"* to not apply process mining in presence of multiple organizations [16], thus using distributed event logs, the techniques already

A. Augusto et al. (Eds.): BPMDS 2022/EMMSAD 2022, LNBIP 450, pp. 63–78, 2022.
https://doi.org/10.1007/978-3-031-07475-2_5

available consider mostly the point of view of a single organization, focusing on (re-)discovery of individual business processes from a single log source [19]. Only a few research lines, i.e., cross-, intra-, and inter-organizational process mining, address, albeit marginally, the problem of discovering on a whole the collaborative behavior of the involved parties and their interactions. This results in a lack of techniques for discovering collaborative models and for detecting issues that typically occur in distributed systems. We refer to problems implied by the interplay among control- and message-flow, e.g., pending messages caused by a lack of synchronization, or a deadlock resulting from activities that are stuck waiting for messages [7,8].

To fill the gap discussed above, we propose a novel **technique for discovering a collaboration model from a set of event logs of a distributed system**. The technique adopts BPMN [14] *collaborations* as target notation, since they provide a suitable modeling abstraction where different organizations exchange messages. It consists of four phases: (i) *logging*, where each system participant locally logs events related to its process execution; (ii) *processes discovery*, producing a process model for each participant using a given discovery algorithm; (iii) *messages analysis*, extracting information suitable to generate the collaboration diagram and to provide analytics on messages exchange; (iv) *collaboration building*, generating a BPMN collaboration model as a combination of the process models and tailoring it to consider distinctive collaboration aspects related to communication. Notably, the technique is parametric to the algorithm used for processes discovery. This allows exploiting algorithms already validated and their reliable implementations defined by the process mining community.

We call COLLIERY (COLLaboration dIscovERY) the technique described above. To foster its adoption, we propose a tool that supports the COLLIERY's phases. The feasibility of COLLIERY has been evaluated in several experiments via logs we produced using a log generator tool, which is a by-product of this work that we also make available.

The rest of the paper is organized as follows. Section 2 provides a running example and introduces the COLLIERY technique. Section 3 presents the related tool. Section 4 reports on the technique evaluation. Section 5 reviews related works. Finally, Sect. 6 concludes and discusses directions for future work.

2 The COLLIERY Technique

This section introduces the BPMN collaboration representing a collaborative scenario used for better presenting the COLLIERY technique and its phases.

The collaboration model in Fig. 1 illustrates a healthcare scenario combining the activities of a *Patient*, a *Gynecologist*, a *Laboratory*, and a *Hospital* as follows. The *Patient* provides details about his/her health status and waits for information related to the home treatment or to hospitalization. The *Gynecologist* coordinates the activities of the *Laboratory* and *Hospital*, caring of blood analysis and hospitalization respectively. The collaboration starts when the *Patient* sends the information about the disease to the *Gynecologist*. Then, the *Gynecologist*

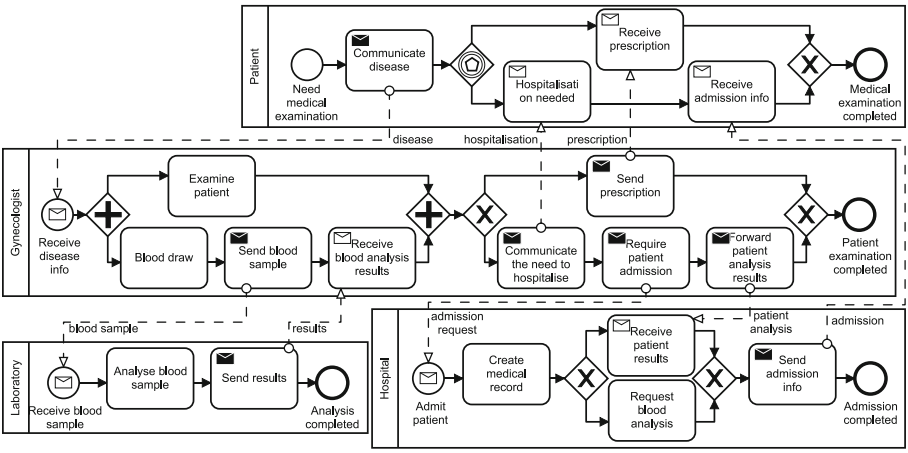

Fig. 1. A healthcare business process collaboration.

examines the *Patient* and, in parallel, draws a blood sample and sends it to the *Laboratory*. The *Laboratory* analyzes the sample and gives back the results to the *Gynecologist*. When both the *Patient* has been examined and the analysis results are received, the *Gynecologist* decides whether to send a medicine prescription or hospitalize the *Patient*, and informs the *Patient* accordingly. Only in the latter case, the *Gynecologist* triggers the *Hospital* by requesting the *Patient* admission and sending the analysis results. When the *Hospital* starts its process, it creates a medical record for the *Patient*, and then decides whether to consider the results of the blood analysis already done or ask for a new analysis; in any case, then it sends the admission information to the *Patient*.

Distributed systems like that can be discovered with the COLLIERY technique we are going to introduce. Figure 2 depicts the structure of the technique, highlighting the phases by which it is composed.

Logging Phase. Process mining relies on the assumption that systems record events about the actual execution of their processes. These events are collected in the so-called *(event) logs*. A log consists of a set of *cases*, each of which refers to a list of events, i.e., a possible run of the system. An *event* refers to the execution of system activity and is described by a set of *attributes*, e.g., the activity name and the timestamp. The sequence of events related to a given case is called *trace*.

The COLLIERY technique relies on logs as well. However, since its goal is to extract information from distributed systems, it has to work on sets of logs. We call *process log* the log of a single participant of a distributed system, and *collaboration log* the set of process logs of all participants of a system. Collaboration logs have the following distinctive features. Firstly, the process logs included in a collaboration log register information about the messages exchanged via communication activities. For example, in our running scenario, an event corresponding to the execution of the activity "Communicate disease" by the *Patient* keeps trace of the sending of a message of type "disease". Secondly, a run of the

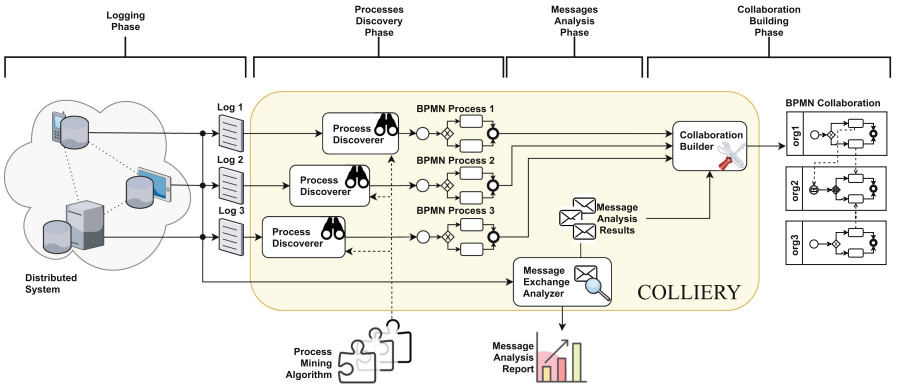

Fig. 2. The COLLIERY technique.

distributed system, namely a *collaboration case*, corresponds to a set of cases one for each involved participant. Figure 3 shows an excerpt of the collaboration log of our running example, where we highlighted the events belonging to two different collaboration cases. In particular, we considered a case of a patient that has been hospitalized (events in blue bounded by a solid line), and a case of a patient that did not need hospitalization and directly received a prescription from the Gynecologist (events in yellow bounded by a dotted line).

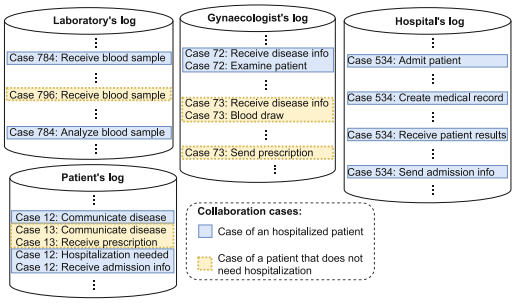

Fig. 3. Example of collaboration cases.

The logging activities of participants are kept independent to ensure the loose coupling of system participants, which is a typical requirement of distributed systems. Hence, we do not rely on any identifier for collaboration cases, which would have required an agreement among the participants. Although the content of each process log is independently produced, events stored in different logs belonging to the same collaboration case may have causal dependencies, which are indeed determined by the exchanged messages through their content. Our technique correlates the collaboration cases assuming the presence of the same *message instance identifier* among the attributes of the sending and receiving events as already done in [10]. This is not a limitation of the approach since unique message identifiers are already applied in several communication protocols, e.g., web-service addressing and HTTP cookie.

Like almost all process mining techniques and tools, we consider event logs compliant with the eXtensible Event Stream (XES) format [12], which is the standard for storing and exchanging event logs. To keep track of the additional

```
<event>
 <string key='communicationMode' value='receive'/>
 <string key='concept:name' value='Receive disease info'/>
 <string key='org:group' value='Gynecologist'/>
 <date key='time:timestamp' value='2021/10/30 11:22'/>
 <string key='msgType' value='disease'/>
 <string key='msgInstanceID' value='disease_38'/>
 <string key='eventType' value='start'/>
</event>
```

Fig. 4. An event with message in XES format.

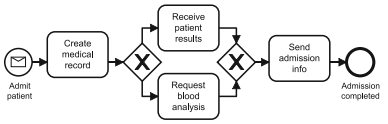

Fig. 5. Hospital's process discovered by the Split Miner.

information about messages and event types required by the COLLIERY technique, we have extended the log format by relying on the extensibility mechanism of XES. Figure 4 shows an example of an extended event drawn from the gynecologist's XES log. This is a *receive* event (key `communicationMode`), corresponding to the system activity *Receive disease info* (key `concept:name`) performed by the *Gynecologist* participant (key `org:group`), who has received on October 30, 2021 at 11:22 (key `time:timestamp`) the message of type *disease* (key `msgType`) uniquely identified by *disease_38* (key `msgInstanceID`); the key `eventType` indicates that this event corresponds to the starting event of the enclosing case for the gynecologist's log. Notably, we assume an asynchronous communication model with point-to-point interactions, meaning that the delivered messages are inserted into queues, and for each message, there is exactly one sender and one receiver.

Processes Discovery Phase. This phase has been specifically designed to exploit process discovery algorithms already defined, and possibly implemented, by the research community. It takes as input a collaboration log under consideration, and generates the corresponding BPMN processes. The models' generation can be realized by means of any process discovery algorithm that produces process models in the BPMN notation, or in other notations that can be automatically translated into BPMN [4]. At the time being, we considered the following algorithms as instantiations for this parameter in our experimentation: Alpha [18], Alpha+ [18], Heuristic Miner [20], Inductive Miner [13], and Split Miner [3]. As a matter of example, by applying the Split Miner algorithm to the Hospital's log of our running scenario, we obtain the BPMN process in Fig. 5. The process is similar to the one enclosed on the Hospital pool in Fig. 1, except for the communication aspects that are not dealt with by the Split Miner algorithm.

Messages Analysis Phase. In this phase, the messages exchange analyzer inspects all process logs to correlate the information concerning the sent messages with the received ones. The aim of this phase is twofold. Firstly, it produces information on communication aspects necessary in the next phase to properly build a BPMN collaboration diagram from the discovered processes. Secondly, analytics on messages delivery and consumption, and related time, are produced to help the user to identify potential issues affecting the proper functioning of the distributed system under analysis.

Let us first focus on the information used for building the collaboration. In the following, we will use a, a_1, a_2, ... to denote activity names (which

correspond to the values of key `concept:name` within an *event* element in the XES logs), m, m_1, m_2, ... to denote message flow names (which correspond to the values of key `msgType` in the XES logs) and o, o_1, o_2, ... to denote organization names (which correspond to the values of key `org:group` in the XES logs). The information passed to the collaboration builder contains firstly a set M of quintuples of the form (o_1, a_1, m, o_2, a_2), meaning that the send activity a_1 in the pool of organization o_1 has to be linked to the receive activity a_2 in the pool of organization o_2 by means of a message flow labeled by the message name m. In addition, the collaboration builder receives the set L of message flows in which one or more messages have been lost (i.e., messages that are sent but not consumed), the set S of activities corresponding to starting events (identified by the value *start* for the `eventType` key), and the predicate $rc(m_1, m_2)$ that holds if m_1 and m_2 are in race condition. More specifically, $rc(m_1, m_2)$ holds if m_1 and m_2 are both sent in the same trace and only the message sent for first is consumed by a receiving event.

Let us consider now the analysis of message exchanges in the collaboration log performed to obtain analytics for the user. For each type of message (a.k.a. message flow in the BPMN model) we compute the following information: *(i)* number of sent and lost messages; and *(ii)* minimum, average, and maximum stay time in the queue corresponding to the message type. The information *(i)* is simply computed by counting the number of sending and receiving events in the logs for a given value of the `msgType` key. The information *(ii)*, instead, requires taking care of the timestamp of events and properly determining the amount of time elapsed between the sending and the corresponding receiving events. For the sake of simplicity, we assume as irrelevant the transmission time (i.e., the amount of time from inserting the message in the queue), and we do not consider clocks de-synchronization issues, i.e. we assume that logs are generated by systems relying on a clock synchronization solution (see, e.g., [15, Ch. 6]) or working in contexts where the clock drift effects are irrelevant. Even if this analysis is not particularly sophisticated, the produced results may be very effective in identifying communication-related issues in the considered system. The analysis results are visualized in intuitive charts to facilitate both quick interpretation and deep analysis. It is worth noticing that, differently from the message exchange analysis required by the collaboration building, this part of the analysis could be extended or customized according to specific user requirements. The messages analysis for our running example identifies that there are some lost messages. This information would allow the user to intervene in the system to fix the issues causing the loss of messages. We discuss the results of this analysis in Sect. 4.

Collaboration Building Phase. The last phase of the COLLIERY technique concerns the building of the BPMN collaboration from the products of the previous phases. Firstly, we enclose each BPMN process discovered in the second phase within a pool element, whose name corresponds to the system participant that has generated the log (recorded in the key `org:group`). At this point, we have a collaboration with disconnected pools, whose processes only include non-communicating activities. For example, given the processes and the set of

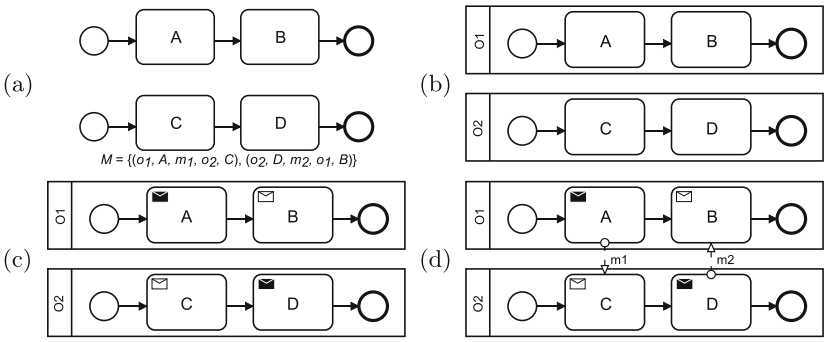

Fig. 6. Collaboration building example.

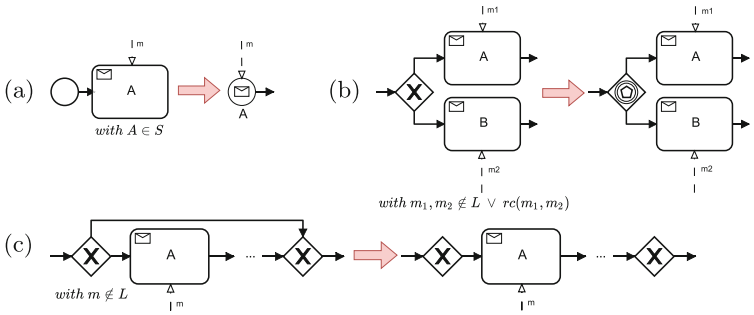

Fig. 7. Fixing communication aspects.

quintuples in Fig. 6(a), the collaboration resulting after these initial operations is the one in Fig. 6(b). Then, send and receive activities in all processes are identified, and hence properly specified in the model. These data can be easily retrieved from the set M of quintuples produced in the third phase: the set of sending activities for an organization o is $\{a \mid (o, a, m, o_2, a_2) \in M\}$, while the set of receiving activities is $\{a \mid (o_1, a_1, m, o, a) \in M\}$. For example, using the quintuples in Fig. 6(a), we obtain the model with specialized activities in Fig. 6(c). Finally, the communicating activities are connected through message flows: for each quintuple (o_1, a_1, m, o_2, a_2) in M, it is inserted in the collaboration model a message flow labeled by m starting from the activity a_1 in the pool o_1 and ending in the activity a_2 in the pool o_2. The final result for the considered simple example is the model in Fig. 6(d).

Since the used process discovery algorithms disregard communication events, the collaboration models obtained so far may present issues. Therefore, a second step in the collaboration building phase is needed to refine the model and properly represent communication aspects. Figure 7 reports the transformation we apply to fix the communication issues. The first transformation, Fig. 7(a), replaces a receive task at the beginning of a process, corresponding to a start event in the log (condition $A \in S$, where S is the set of starting activities computed in the third

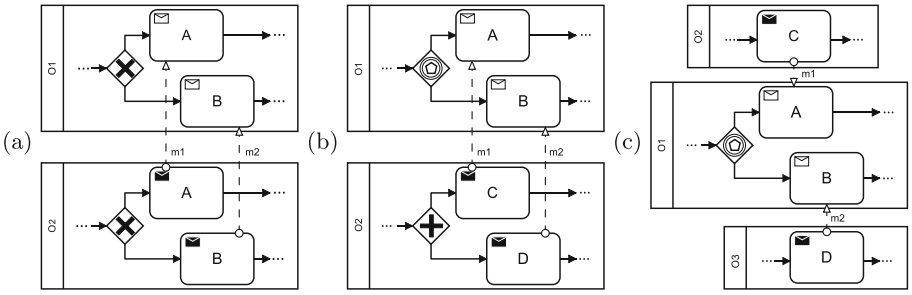

Fig. 8. Examples of model fragments with (a) lost messages, (b) race condition due to parallel gateway, (c) race condition due to multiple sender participants.

phase), by a BPMN message start event element. In this way, in the execution of the resulting BPMN collaboration model, the considered process will be instantiated and started only when a message of type m is actually received. The second transformation, Fig. 7(b), replaces an exclusive choice (realized in BPMN by means of an XOR gateway) between receive activities by a message-driven choice (realized in BPMN by means of an event-based gateway). The figure depicts only two activities, but the transformation works likewise with more than two activities. This transformation is applied when either *(i)* there are no lost messages for the involved receives (condition $m_1, m_2 \notin L$, where L is the set, computed in the third phase, of message flows that have lost messages) or *(ii)* there is a race condition from the messages incoming into the involved receives (condition $rc(m_1, m_2)$, where rc is the race condition predicate computed in the third phase). In fact, the event-based gateway may not be appropriate when the condition *(i)* is not satisfied, because the gateway permits to receive any type of message between m_1 and m_2. Instead, as in the example in Fig. 8(a), the use of an XOR gateway can lead to situations where a participant is waiting for a given message, say m_1, while another is sending another type of message, say m_2; in such a case, the message of type m_1 will be lost ($m_1 \in L$). Notably, this transformation is a heuristic rule; in fact, there may be issues (e.g., a deadlock upstream) causing the loss of messages. Condition *(ii)* permits, instead, to apply the transformation also in some cases of lost messages. In case of a race condition between messages (as in the examples in Fig. 8(b)–(c)), the first arrived message triggers the corresponding receiving activity and disables the others, hence the other messages will be lost. In these situations, the event-based gateway is the appropriate gateway, as it is the BPMN element specifically devoted to dealing with race conditions. Finally, the third transformation, Fig. 7(c), aims at fixing misbehavior concerning the blocking capability of receive activities. Indeed, process mining algorithms do not distinguish between sending, receiving, and internal activities, hence it considers all of them as non-blocking elements. However, when asynchronous communication enters the game, a receive activity has to wait for the corresponding message, possibly forever. Consider, for example, a collaboration log composed of: the process

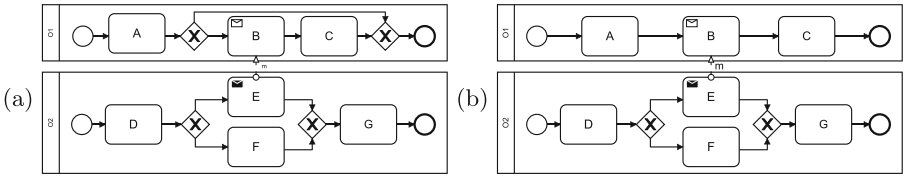

Fig. 9. Examples of discovered collaboration (a) before and (b) after the transformation in Fig. 7(c).

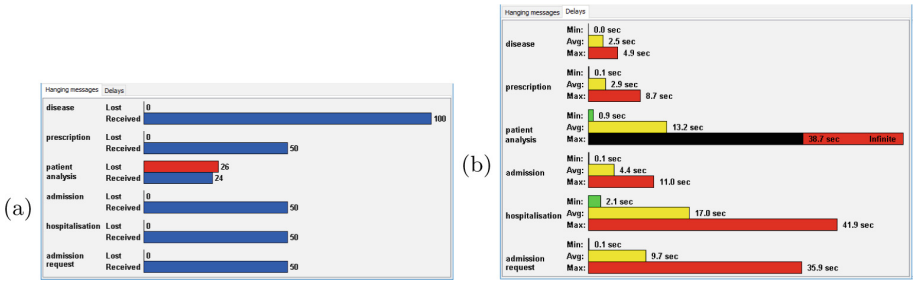

Fig. 10. COLLIERY interfaces for (a) pending messages and (b) message delays.

log of the organization o_1, containing h occurrences of trace $\langle ABC \rangle$ (with B receiving a message m) and k occurrences of trace $\langle A \rangle$, and the process log of the organization o_2, containing h occurrences of trace $\langle DEG \rangle$ (with E sending a message m) and k occurrences of trace $\langle DFG \rangle$. In this case, every mining algorithm properly discovers the process corresponding to the o_2's log, while for the o_1's log the model may differ: the Alpha and the Alpha+ generate a process producing only $\langle ABC \rangle$ traces, the Inductive generates an overfitting model, and only the Heuristic and the Split miners properly discover the process. For instance, the collaboration in Fig. 9(a) has been discovered with the Split Miner. In this case, the coexistence of traces $\langle ABC \rangle$ and $\langle A \rangle$ has been interpreted as the possibility of skipping activities B and C after the execution of A, which would be a correct interpretation if one did not take into account the blocking behavior of the receive activity in a communicative scenario. However, this collaboration model does not faithfully represent the behavior registered in the collaboration log, because it allows execution traces where activity E is performed while activities B and C are skipped, leading to losses of messages of type m that do not occur in the log. Instead, the collaboration in Fig. 9(b), resulting from the application of the transformation in Fig. 7(c), does not exhibit this issue, but it precisely represents the content of the collaboration log.

3 The COLLIERY Tool

We present here the COLLIERY tool implementing three out of the four phases of the technique presented in Fig. 2, since the Logging phase is charged to the

distributed system itself. The COLLIERY tool takes as input a set of logs of different organizations and generates a BPMN collaboration model along with a communication analysis report. The tool is developed in *Java*, to guarantee compatibility with any operating system, and exploits external libraries, which helped us to implement the COLLIERY's phases. For the Processes discovery phase we make use of the open-source platform PM4Py (https://pm4py.fit.fraunhofer.de/), as it implements many discovery algorithms, i.e., Alpha, Alpha+, Inductive Miner, and Heuristics Miner, and the transformation algorithm to obtain BPMN models. For the Split Miner we used the Java implementation introduced in [3]. In this way we decouple the process discovery functionalities from the rest of the tool, thus supporting the integration of different discovery algorithms and mining tools. Notably, COLLIERY allows (where applicable) the specification of parameters influencing the discovery algorithms (e.g., the dependency threshold for the Heuristic algorithm). For the Messages analysis phase, the COLLIERY tool parses and manages the input XES files using OpenXES. Finally, the Collaboration building phase uses the Camunda API to generate a fresh collaboration diagram on which to insert the discovered processes and decorate the elements. The COLLIERY tool is provided as a runnable *jar* file; the binary files and the source code, instructions, and examples are available at https://pros.unicam.it/colliery/. Part of its graphical interface is shown in Fig. 10(a)–(b).

4 Experimental Evaluation

This section presents the technique evaluation carried out with the tool on a set of scenarios, including the running example, to check the quality of the discovered collaborations, and to discuss the outcomes of the communication analysis.

Dataset. The following experiments have been conducted on ten collaboration logs representing the executions of distributed systems. Usually, real(-istic) event logs are made available by open-access repositories (e.g., https://data.4tu.nl), or are synthetically generated by tools (e.g., https://plg.processmining.it). In both cases, the logs that can be obtained represent the executions of single organization processes in which the communication events are missing. Therefore, we developed a new tool for logs generation, which is a by-product of this work that we also make available. It executes BPMN collaborations and records activity and message events into XES files, one for each participant, as discussed in Sect. 2. In addition to the running example, we selected artificial and realistic collaboration models, and we generated collaboration logs from their execution. Notably, the models we selected differ in: the number of participants (from 2 to 4), size (from 16 to 42), and the number of messages (from 2 to 8). Moreover, for making the dataset as heterogeneous as possible, some models are unsound, unsafe, unstructured, or contain loops. The generator of collaboration logs, the models, the event logs, and all the data used and produced in the evaluation are made available at http://pros.unicam.it/colliery/.

Evaluation Approach. Discovery techniques are evaluated through conformance checking [17]: it assesses the quality of a discovered model by comparing

Table 1. Results of the evaluation (f stands for fitness, and p for precision).

	Artificial 1		Artificial 2		Artificial 3		Artificial 4		Artificial 5	
	f	p	f	p	f	p	f	p	f	p
alpha	1	0.7242	0.9261	0.6348	0.9573	0.4565	1	0.6050	0.9149	0.3973
alpha+	1	0.7242	0.9350	0.6378	0.9573	0.4714	1	0.6050	0.9149	0.3973
heu.	1	0.7242	0.9350	0.6378	0.9573	0.4133	1	0.6050	0.9149	0.3973
ind.	1	0.7242	0.9350	0.6378	0.9460	0.4457	1	0.6050	0.9149	0.3973
split	1	0.7242	0.9605	0.6121	0.9563	0.3865	1	0.6050	0.9149	0.3973
	Real 1		Real 2		Real 3		Real 4		Real 5	
	f	p	f	p	f	p	f	p	f	p
alpha	unb.	unb.	unb.	unb.	unb.	unb.	unb.	unb.	unb.	unb.
alpha+	unb.	unb.	unb.	unb.	unb.	unb.	unb.	unb.	unb.	unb.
heu.	unb.	unb.	0.7867	0.6613	unb.	unb.	0.9999	0.7593	unb.	unb.
ind.	0.7913	0.6629	0.8753	0.3832	0.9663	0.2729	0.9945	0.7866	0.7786	0.7043
split	0.7457	0.6176	unb.	unb.	unb.	unb.	0.9220	0.7570	0.7437	0.6863

the behavior observed in an event log with the one described by a process model. Unfortunately, the conformance checking techniques and tools available today compare process models (usually Petri nets) with process logs while lacking approaches that compare collaboration models and collaboration logs.

To apply conformance checking in our context it is necessary to transform each discovered collaboration into a Petri net, and appropriately merge the traces of each participant into a *collective* event log, i.e., a single log file where the traces contain ordered lists of events triggered by any participant. Concerning the collective logs, they are generated by the above-introduced log generator tool we developed. Instead, the translations of the BPMN collaborations into behaviorally equivalent Petri nets have been performed in two steps. The first step consists of using the *Convert BPMN diagram to Petri net (control-flow)* plugin of ProM (www.promtools.org) to produce a set of Petri nets, each of which represents the control-flow of a participant process. In the second step, we combine these Petri net processes to include also the message-flow. This is achieved by connecting through a place each transition that represents a sending action to the transition that represents the corresponding receiving activity. Therefore, with such data and the aid of ProM, we performed conformance checking to measure *fitness*, i.e., the ability of a model to reproduce the behavior contained in a log, and *precision*, i.e., the ability of a model to generate only the behavior discovered in a log, following respectively the approaches proposed in [1] and [2].

Evaluation Results. Hereafter, we present the result of the experiments. For the sake of presentation, we discuss in detail only the results obtained on the running example. Independently from the mining algorithm selected for the discovery phase, the collaboration models obtained using COLLIERY on the collaborative log of the running example report the four pools of the original model

(Fig. 1), correctly labeled with the corresponding organization name. All the message flows, the event-based gateway in the *Patient* pool, and the message start events in the *Hospital*, *Laboratory* and *Gynecologist* pools are discovered. Moreover, referring to the collaboration discovered by COLLIERY with the Inductive Miner algorithm, also the participant processes are identical in the topology to the original ones, while the other algorithms fail in reproducing properly the block of parallel tasks. For instance, with the Heuristics Miner this block of tasks ends with an exclusive join gateway (instead of a parallel one), and the task *Receive blood analysis results* is placed after the block. This discrepancy changes the behavior of the whole system, introducing unsafeness. This is not due to the Collaboration building phase; it is due to the discovery of the *Gynecologist* process made by the Heuristics Miner. In fact, considering a different discovery algorithm we obtain a different result. In particular, by selecting the Inductive Miner, the *Gynecologist* process results identical to the original model (see the repository we made available online).

About the results of the conformance checking, only the collaborations of the running example discovered using the Inductive and Split Miner can be analyzed, as the others cannot be transformed into bounded Petri nets. The collaboration discovered by means of the Inductive Miner has the higher results (fitness $\approx$0.79 and precision $\approx$0.66), strictly followed by the results of the Split Miner (fitness $\approx$ 0.75 and precision $\approx$ 0.62). In both cases, the values of the conformance checking show that the collaborations discovered by COLLIERY are good in reproducing the behaviors shown in the logs without overfitting them too much. The values of fitness and precision achieved for the other examples with different discovery algorithms are resumed in Table 1. The first five rows regard the collaborations discovered from event logs of artificial (and structured) models. In this case, we can always calculate fitness and precision values because all the collaborations discovered by COLLIERY are bounded. While the last five rows regard collaboration discovered from real (and often unstructured) models that in fact result very often in unbounded nets for which we cannot apply conformance checking. Overall, the observed values are high, especially for the fitness that reaches in some cases the maximum (i.e., 1).

Regarding the communication analysis performed on the event logs of the running example, Fig. 10(a) reports the number of messages exchanged or lost for each message flow name. From this plot, we can observe a problem with messages of type *patient analysis*. Specifically, 26 messages have been sent but not received, while 24 have been correctly received. This information permits to spot a potential problem in the distributed system under analysis, whose identification is facilitated by the discovered model that provides an abstract view of the system behavior. Indeed, the *Gynecologist* always forwards the patient analysis to the *Hospital* that, in its turn, can skip the receive task *Receive patient analysis*. Figure 10(b) reports the minimum, the maximum, and the average number of seconds elapsed between a send event and a receive event with the same message instance identifier. Notably, in the case of lost messages, the tool depicts a maximum time equal to infinite, together with the maximum time calculated

considering only the received messages. This information opens the possibility to monitor and predict the delays related to message exchanges, thus enabling the identification of bottlenecks.

Threats to Validity. Since process mining focuses almost entirely on process logs rather than on collaboration ones, datasets and approaches supporting the evaluation of techniques like COLLIERY are missing, as also reported in [9]. A possible solution would be to transform existing event logs into distributed logs suitable for our technique. However, this would imply manually inserting communication events, thus knowing the system that generates these logs. Another concern regards the absence of conformance checking approaches and related tools supporting the evaluation of discovery techniques for collaborative scenarios. We managed to arrange collaboration logs and BPMN collaborations to work with existing conformance checking approaches, but a conformance technique specific for this collaborative setting would avoid or reduce errors introduced by logs and models transformations. Indeed, despite very often COLLIERY discovers exactly the original model, fitness and precision values are lower than 1.

5 Related Works

Despite almost all process mining approaches being devoted to gathering knowledge on single organization processes, works exploiting process mining in collaborative settings exist in the literature. These techniques come under the umbrella of *cross-*, *intra-*, and *inter-organizational* process mining. Cross-organizational process mining aims at spotting differences between processes of the same or different organizations [17]. Intra-organizational process mining tends to detect resources, roles, and departments involved in single organization processes [23]. While, more in line with our work, inter-organizational process mining deals with logs distributed over different organizations [10,16]. Here we discuss approaches somehow similar to ours.

Zeng et al. present in [22] a framework for the discovery of cross-organizational models, where participants can communicate. The framework relies on distributed logs, each of which permits the discovery of a colored Petri-net enriched with resources and communication. Then, these nets are grouped in a collaborative workflow via coordination patterns. Differently from us, this approach does not allow the selection of the desired discovery algorithm. Moreover, the use of Petri-nets, instead of BPMN collaborations, results in a less intuitive modeling notation, reducing the comprehension of the system behavior significantly. Finally, the approach does not give insights about the message exchange, and no tool support is given. Bernardi et al. define a similar approach in [5] resulting in the discovery of business rules, instead of models. In the same fashion, Zeng et al. provide in [21] an approach for building Petri-nets from distributed event logs. The approach produces a top-level process model enriched with abstract transitions representing coordination models among the participants. Every abstract transition refers to a single participant process given

as output using standard Petri-nets. Finally, the participant processes are integrated with the coordination model obtaining the whole collaboration. Even this approach has not been implemented in a tool, and no support to high-level notation, like BPMN, and no insights on the communication are given. Differently, Engel et al. [10] present a framework addressing the inter-organizational process mining of organizations interacting via the Electronic Data Interchange (EDI) messages standard. The framework permits getting insights from EDI data by transforming them into event logs. However, the focus of this work is more on extracting information about the interactions, than on producing collaboration models. Indeed, on the produced logs, the authors apply existing discovery algorithms that cannot produce collaborations. Hernandez-Resendiz et al. present in [11] a methodology to discover choreographies from the logs of distributed organizations. The methodology merges the logs on the basis of a similarity matrix obtained by calculating the distance between the traces of each participant and discovers the choreography by means of the Split Miner. Differently from us, no automatic tool is provided, the possibility to use other discovery algorithms is forbidden, and the number of participants is limited to two. Finally, Elkoumy et al. show in [9] an approach for applying process mining in collaborative scenarios without exposing sensitive data, business secrets, etc. The approach makes the organizations' logs anonymous and extracts from them a directly-follows graph, to which apply the discovery. This work points out the security problems that may arise when we deal with data from different organizations. Despite our technique does not address this problem, we could easily extend it to preserve privacy: the Processes discovery phase could be performed internally to each organization, while the Messages analysis phase can be performed in the same way on logs that have been anonymized.

6 Concluding Remarks

This paper presents COLLIERY: a technique for discovering collaboration models from distributed event logs. COLLIERY exploits existing discovery algorithms to re-create process models of organizations involved in a distributed system, then it merges them into a BPMN collaboration. The resulting model is decorated in order to reproduce the communication aspects extrapolated from the logs. Moreover, COLLIERY provides an analysis of the communication events to get insights about message exchanges. Finally, COLLIERY has been implemented in a tool we used to evaluate the technique against several logs.

Discussion. We were motivated by the increasing adoption of distributed paradigms in IT systems and by a general lack of process mining solutions suitable for these scenarios. In particular, almost all the discovery techniques consider the perspective of single organizations. Driving process mining to deal with distributed scenarios can bring the advantage of gathering information on message exchanges and on their impact on the involved processes. The technique we propose could have practical applicability in many research fields around which

the BPM and the process mining communities are spending a lot of efforts such as the Internet of Things, Cyber-physical systems, and microservices, in which the distribution of the information is even more evident.

Future Work. We plan to implement the COLLIERY technique within existing process mining frameworks, e.g., ProM. On the one hand, this would allow increasing the number of supported discovery algorithms. On the other hand, researchers would have the possibility to develop related techniques such as conformance checking or model enhancement suitable for collaborations. Moreover, we plan to support other methods for correlating the collaboration cases to make COLLIERY works also in case message identifiers are not present, for instance, using pattern matching or other heuristics on the attributes contained in the message events.

References

1. Adriansyah, A., van Dongen, B., van der Aalst, W.: Conformance checking using cost-based fitness analysis. In: Enterprise Distributed Object Computing, pp. 55–64. IEEE (2011)
2. Adriansyah, A., Munoz-Gama, J., Carmona, J., van Dongen, B.F., van der Aalst, W.M.P.: Alignment based precision checking. In: La Rosa, M., Soffer, P. (eds.) BPM 2012. LNBIP, vol. 132, pp. 137–149. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36285-9_15
3. Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Polyvyanyy, A.: Split miner: automated discovery of accurate and simple business process models from event logs. Knowl. Inf. Syst. **59**(2), 251–284 (2018). https://doi.org/10.1007/s10115-018-1214-x
4. Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Bruno, G.: Automated discovery of structured process models from event logs: the discover-and-structure approach. Data Knowl. Eng. **117**, 373–392 (2018)
5. Bernardi, M.L., Cimitile, M., Mercaldo, F.: Cross-organisational process mining in cloud environments. Inf. Knowl. Manag. **17**(02), 1850014 (2018)
6. Beschastnikh, I., Brun, Y., Ernst, M., Krishnamurthy, A.: Inferring models of concurrent systems from logs of their behavior with CSight. In: International Conference on Software Engineering, pp. 468–479. ACM (2014)
7. Corradini, F., Morichetta, A., Polini, A., Re, B., Rossi, L., Tiezzi, F.: Correctness checking for BPMN collaborations with sub-processes. Syst. Softw. **166**, 110594 (2020)
8. Corradini, F., Muzi, C., Re, B., Rossi, L., Tiezzi, F.: Formalising and animating multiple instances in BPMN collaborations. Inf. Syst. **103**, 101459 (2022)
9. Elkoumy, G., Fahrenkrog-Petersen, S.A., Dumas, M., Laud, P., Pankova, A., Weidlich, M.: Secure multi-party computation for inter-organizational process mining. In: Nurcan, S., Reinhartz-Berger, I., Soffer, P., Zdravkovic, J. (eds.) BPMDS/EMMSAD -2020. LNBIP, vol. 387, pp. 166–181. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49418-6_11
10. Engel, R., et al.: Analyzing inter-organizational business processes. IseB **14**(3), 577–612 (2015). https://doi.org/10.1007/s10257-015-0295-2

11. Hernandez-Resendiz, J.D., Tello-Leal, E., Marin-Castro, H.M., Ramirez-Alcocer, U.M., Mata-Torres, J.A.: Merging event logs for inter-organizational process mining. In: Zapata-Cortes, J.A., Alor-Hernández, G., Sánchez-Ramírez, C., García-Alcaraz, J.L. (eds.) New Perspectives on Enterprise Decision-Making Applying Artificial Intelligence Techniques. SCI, vol. 966, pp. 3–26. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-71115-3_1
12. IEEE: Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams (2016)
13. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs containing infrequent behaviour. In: Lohmann, N., Song, M., Wohed, P. (eds.) BPM 2013. LNBIP, vol. 171, pp. 66–78. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-06257-0_6
14. OMG: Business Process Model and Notation (BPMN) v2.0 (2011)
15. Tanenbaum, A., van Steen, M.: Distributed Systems. Pearson, London (2007)
16. Aalst, W.M.P.: Intra- and inter-organizational process mining: discovering processes within and between organizations. In: Johannesson, P., Krogstie, J., Opdahl, A.L. (eds.) PoEM 2011. LNBIP, vol. 92, pp. 1–11. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24849-8_1
17. van der Aalst, W.: Process Mining: Data Science in Action. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49851-4
18. van der Aalst, W., Weijters, T., Maruster, L.: Workflow mining: discovering process models from event logs. IEEE Trans. Knowl. Data Eng. **16**(9), 1128–1142 (2004)
19. Vom Brocke, J., Rosemann, M.: Handbook on Business Process Management 1. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-45100-3
20. Weijters, A., van Der Aalst, W., De Medeiros, A.: Process mining with the heuristics miner-algorithm. TU/e, Technical report, WP 166, pp. 1–34 (2006)
21. Zeng, Q., Duan, H., Liu, C.: Top-down process mining from multi-source running logs based on refinement of petri nets. IEEE Access **8**, 61355–61369 (2020)
22. Zeng, Q., Sun, S.X., Duan, H., Liu, C., Wang, H.: Cross-organizational collaborative workflow mining from a multi-source log. Decis. Support Syst. **54**, 1280–1301 (2013)
23. Zhao, W., Zhao, X.: Process mining from the organizational perspective. In: Wen, Z., Li, T. (eds.) Foundations of Intelligent Systems. AISC, vol. 277, pp. 701–708. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54924-3_66