# Monitoring Business Process Interaction

Nico Herzberg, Matthias Kunze, and Mathias Weske

Hasso Plattner Institute at the University of Potsdam
Prof.-Dr.-Helmert-Strasse 2-3, 14482 Potsdam
{nico.herzberg,matthias.kunze,mathias.weske}@hpi.uni-potsdam.de

**Abstract.** Business process monitoring provides well established means to track the history and state of business processes and to evaluate their performance. However, common techniques and systems lack support of monitoring processes and their interactions with partners, particularly if such processes are not automated or only few events can be detected, rather than a complete log.

In this paper, we present a mechanism to determine the state of business processes with only few events to be detected. Based on this mechanism, we provide a formal framework to monitor processes and their interaction with collaboration partner's processes, as well as to evaluate the processes' performance. The framework has been applied in the context of an industry project, which we used to evaluate our solution.

## 1 Introduction

Business process management has received considerable interest among modern companies, as it sustains competitiveness in an ever-changing market environment. Organizations capture their operations in business process models for documentation, but also for automation and certification. Evaluating business processes, in particular with respect to execution performance, requires means to monitor the state of a process instance by tracking business relevant events. Business process monitoring is a well-established discipline, and has received considerable support in modern business process management systems [6]. These systems automatically orchestrate a business process, i.e., a workflow engine enacts the technically configured process, and hence knows about the start and end of every activity and decision taken. This information is called a process log and can be used to track the state and history of running and terminated business processes [15].

However, a large share of business processes is enacted in a non-automated manner, i.e., they are not executed by means of a workflow engine but conducted manually by humans following certain guidelines. Consequently, there is no central record of conducted activities comparable to a process log, but only few activities, events, or side effects thereof can be observed in the technical environment of the business process. For instance, the process of creation, review, and iterative update of a document may only be discovered by revision information in a document management system.

Moreover, many processes involve collaboration beyond the boundaries of an organization or department. Such collaboration is typically not controlled centrally, as each organization conducts their own process; interaction is carried out by sending and receiving messages, e.g., requesting and providing feedback on a given document. This

A. Ghose et al. (Eds.): ICSOC 2012, LNCS 7759, pp. 228–240, 2013.

makes it difficult to monitor and evaluate the process' execution. However, providing insight into interacting processes to all collaborators increases transparency and suggests to improve collaboration.

In this paper, we address these issues and present a framework to monitor process interaction in absence of a central business process management system. In more detail, we provide a formal framework based on the notion of workflow modules, partner synthesis, and event monitoring points. Even if the log of events is sparse, this allows capturing the state of a process, monitoring process interactions, and detecting whether a partner cannot proceed, because it is waiting for a message from another partner. Based on this framework, we introduce interaction monitoring models, an abstraction of the internal behavior of a process, as a means to monitor the state of a process on a coarsely grained level and to communicate it to collaborators in real time. These models provide instant insight into the current state of a process and allow telling who is currently active, waiting for someone, or responsible for a delay. Additionally, we emphasize how the performance of business processes and process interactions can be evaluated in quantitative terms based on this framework.

The remainder of this paper is structured as follows. We motivate business process interaction monitoring with a case study, to which the proposed framework has been applied, in Section 2. Section 3 provides the formal framework of our approach and explains, how it enables process interaction monitoring and performance evaluation. We then revisit the motivational use case and discuss its implementation in Section 4, before we discuss related work in Section 5, conclude the paper and give a brief outlook on future work, in Section 6.

## 2 Case Study

In this section, we present the case of a German health insurance company, who carried out mergers of several federal branches, recently. These branches independently carried out similar operations for years. However, after the merger, it turned out that different job specifications existed for the same position in the company across the former branches. Hence, the *organization development* (OD) department became in charge of harmonizing these job specifications.

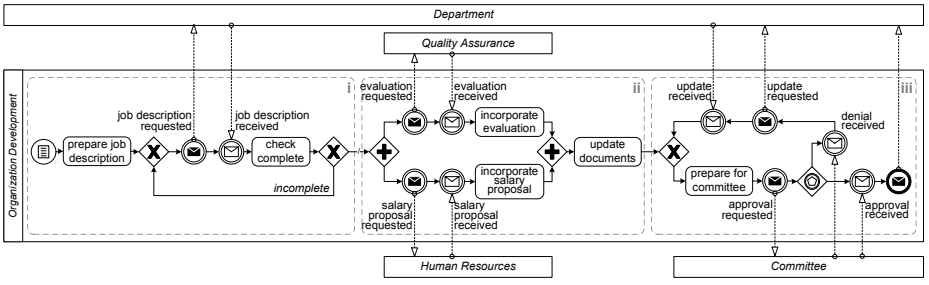


**Fig. 1.** Job specification harmonization process of a health insurance company (simplified model)

Fig. 1 shows the BPMN process model that has been established to harmonize job specifications. It involves several interaction partners: *Quality Assurance*, *Human Resources* (HR), a *Department* that embraces the position to be harmonized, e.g., Medical Care, Geriatric Care, IT, and HR, and a *Committee* that eventually approves the harmonized job specifications. It is also possible that one department engages in several roles, e.g., if a position in Human Resources needs to be harmonized.

In the process model, we see that the OD department starts the process and prepares the job description documents. Subsequently, it requests a description of the position from the respective Department and checks it for completeness upon reception. Please note, that interaction between the departments has been modeled as message passing between separate pools in the model. If the job description from the Department is incomplete, the Department must improve it in an iterative fashion. Otherwise, it is handed over to Quality Assurance that ensures that the job description meets the company's requirements. At the same time, Human Resources receive the job description with a request to propose a salary for the position. After both inputs have been incorporated, the OD department updates the job specification documents. Eventually, the job specification is prepared and sent to the Committee whose sole task is to review and approve the specification. In case approval is denied, the respective Department must update it, and it needs to be reviewed again by the Committee.

The OD department strives to harmonize a job specification within three months. However, in a fair amount of cases, one or several of the interaction partners impede progress as they do not respond in a reasonable period of time. Hence, the OD department requires a means (a) to track the status of a process instance, i.e., show the current state, show previous actions, and point out, whether the instance is waiting for a response from one partner, (b) to answer who is responsible for a delay, and (c) to provide time measurements about the duration of the complete process, its activities, and how much time respective interaction partners have spent for their contributions.

Nevertheless, the OD department was reluctant to disclose their internal, detailed process model to interaction partners. Therefore, they identified three phases, *job description* (i), *quality assurance* (ii), and *approval* (iii), grouped by dashed boxes in Fig. 1. The information above should then be projected on a simple model of these phases and be presented to the other departments. By that, the OD department offers process monitoring based solely on their knowledge of the process, i.e., the processes of interaction partners are not known, whilst enabling monitoring and performance evaluation of interactions with its partners.

## 3 Interaction Monitoring Framework

Based on the requirements and issues mentioned in Section 2, we first introduce workflow modules to formally capture interacting processes in Section 3.1. Subsequently, we show how processes can be monitored, even if only few events can be detected and how this information is used to provide monitoring views that provide insight into the state and history of a process to collaborators (a, b) in Section 3.2, before we explain how to evaluate process performance based on this model (c) in Section 3.3. We briefly discuss basic assumptions and limitations of our approach in Section 3.4.

### 3.1 Modeling Interactions

To illustrate our approach, we utilize the example process model from Section 2, cf. Fig. 1. Here, we assume that the process along with its interaction partners is captured in a detailed model, whereas the process of the partners is generally unknown. For instance, in BPMN, process interaction is modeled through message events and message flow, and partners can be represented as black box pools.

To formalize the interaction between processes, we resort to workflow modules, which have been introduced in [9]. Workflow modules are essentially Petri nets—a commonly used formalism for execution semantics of business processes [16]—with additional places that represent sent and received messages, referred to as input and output places of a module hereafter.

**Definition 1 (Workflow Module).** *A* workflow module *is a tuple* $N = (P, T, F, P_i, P_o)$*, where* $(P, T, F)$ *is a Petri net that consists of finite disjoints sets of places* $P$ *and transitions* $T$*, and a flow relation* $F \subseteq (P \times T) \cup (T \times P)$*. For* $X = P \cup T$*, we refer to* $\bullet x = \{y \in X | (y, x) \in F\}$ *as the preset and* $x\bullet = \{y \in X | (x, y) \in F\}$ *as the postset of a node* $x \in X$*, respectively.*

$P_i \subseteq P$ *denotes the set of of input places of N, such that* $\forall p \in P_i : \bullet p = \emptyset$*, and* $P_o \subseteq P$ *the set of output places of N, such that* $\forall (p \in P_o) : p\bullet = \emptyset$*. No transition is connected to an input place and an output place at the same time, i.e.,* $\forall p_i \in P_i, p_o \in P_o \nexists t \in T : t \in p_i\bullet \wedge t \in \bullet p_o$*. The state, or* marking*, of* $(P, T, F)$ *is defined by a function* $M : P \rightarrow \mathbb{N}$.

For the given process model and each partner, a workflow module needs to be created. Interaction of the respective partners is modeled by fusing input places of one module with output places of another one, i.e., given the set of workflow modules of a process and its interaction partners, $\mathcal{N}$, it holds that $\forall N \in \mathcal{N}\ P_i^N \subseteq \bigcup_{N' \in \mathcal{N} \setminus \{N\}} P_o^{N'}$, where $P_i^N$ ($P_o^N$) represents the input (output) places of a module $N \in \mathcal{N}$. Hence, if one module puts a token on one of its output places, this token signals the message transmission, because it is at the same time on the input place of another module.

The process model is translated into a workflow net following common Petri net-based formalizations [8], e.g., activities and events are represented by transitions connected through places, diverging (merging) XOR-gateways by a place that has outgoing (incoming) arcs to (from) several transitions, and forking (joining) AND-gateways by a transition that has outgoing (incoming) arcs to (from) several places. Message exchange (dashed arcs in Fig. 1) between the process and a partner is captured by adding input and output places for each incoming and outgoing message arcs [9].

As the processes of the partners are typically not known, they cannot be translated into workflow modules. However, partners that provide compatible behavior to the process can be synthesized from the internal state and the interaction of the process [18]. A synthesized partner describes the observable behavior at the interface between the partner and the process. By that, we can correlate events observed at the interaction interface of the process.

In the context of process monitoring, internal life cycles for activities have been proposed to be captured, when translating models to Petri nets, e.g., [7], where one activity would be translated into a subnet that becomes enabled, running, and terminated. However, in the present case, we are only interested in the completion of activities or

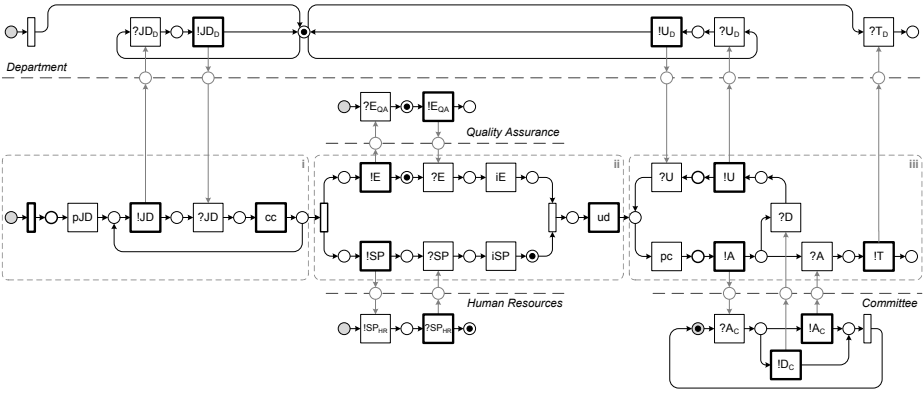


**Fig. 2.** Workflow modules for the job specification harmonization process, cf. Fig. 1. Shaded places represent start places of the modules. The current marking, i.e., distribution of tokens to places, represents a certain state during the execution of one process instance.

interactions. Hence, an activity or event of the process model can safely be represented as a single transition in the workflow module. Consequently, termination of an activity or event is represented by firing its corresponding transition.

Fig. 2 depicts the fused workflow modules for the job specification harmonization process and its interaction partners. The module in the middle of the figure represents the process, which has been translated into a workflow module as presented above, the partner modules (separated by dashed horizontal lines) of the Department, Quality Assurance, Human Resources and the Committee have been synthesized. For brevity, we used acronyms for activities, e.g., *pJD* represents activity "process job description" and *cc* stands for "check complete".

BPMN message events have been translated to transitions, whereas sending a message is marked by a leading "*!*" in the transition label and receiving a message by a leading "*?*". Here, process interaction typically consists of two messages, a request to a partner and the partner's response. For instance, transition *!JD* stands for requesting a job description. This request is received by the Department, represented through transition $?JD_D$, and responded with $!JD_D$. This response is received by the process through *?JD*. The remaining interactions are analogous.

It is important to understand that the workflow modules of interaction partners may differ from their actual processes as these modules only describe the observable behavior of these partners at the interface of the process. One can understand this as part of the process that is not explicitly modeled.

We captured the process of the Committee as a continuous loop in the corresponding module, rather than as one process instance for each review. This is due to the Committee's sole task to review job specifications, i.e., the Committee may review several job harmonization process instances in the course of one session. Nevertheless, the fused Petri net is free from dead locks.

### 3.2 Monitoring Interaction

The workflow modules depicted in Fig. 2 are already marked, i.e., some places contain tokens. Based on a marking, we define the state and execution semantics of workflow modules as follows.

**Definition 2 (Workflow Module Semantics).** *Let $(P, T, F, P_i, P_o)$ be a workflow module and $M$ be a marking.*

- *A transition $t \in T$ is enabled, iff $\forall p \in \bullet t : M(p) \geq 1$.*
- *A transition $t \in T$ is waiting, iff $\forall p \in \bullet t \setminus P_i : M(p) \geq 1 \wedge \forall p \in \bullet t \cap P_i : M(p) = 0$.*
- *The firing of a transition $t$ in state $M$ where $t$ is enabled results in a state $M'$ where $\forall p \in \bullet t : M'(p) = M(p) - 1 \wedge \forall p \in t\bullet : M'(p) = M(p) + 1$*

In Fig. 2, the process instance is in a state where requests for an evaluation of the job specification and for a salary proposal have been sent to Quality Assurance and Human Resources. While Human Resources has already responded with a proposal that has been incorporated into the job specification, i.e., transition *iSP* has fired, the process is waiting for a response from Quality Assurance which has not been received—transition $!E_{QA}$ of Quality Assurance has not fired yet. Since only the token from the input place of *?E* is missing for enablement, transition *?E* is *waiting*.

To monitor a process instance and interaction with its partners, a set of event monitoring points is required, i.e., information that a state has changed. We represent these by special transitions that do not fire until the corresponding event has been observed. Such events need to be detected in or obtained from the environment of the business process, e.g., by regularly testing, whether the state of a document has changed. Also, for many manually conducted processes, a checklist document, e.g., a spread sheet file, is used to record the accomplishment of an activity with its time and date.

In Section 4 we illustrate the architecture of a system, based on earlier work [7], that obtains events, correlates them with process instances, and provides them to event monitoring points. Here, we assume such events have been detected and correlated, already.

**Definition 3 (Event Monitoring Point, Transition Firing).** *Let $(P, T, F, P_i, P_o)$ be a workflow module. An event monitoring point represents a business event that triggers an observable state change of the net, denoted by the firing of a transition $t \in T$. The set of event monitoring points is defined as $T_e \subseteq T$.*

*Firing of $t \in T_e$ is deferred until the event related with $t$ has been monitored; all other transitions, i.e., $t \in T \setminus T_e$, fire immediately, when they become enabled.*

In Fig. 2, event monitoring points are visualized by a bold outline. One can see that transition $!E_{QA}$ is enabled. Since a message from Quality Assurance has not been received yet, i.e., the according event has not been detected, $!E_{QA}$ did not fire. As we discussed in Section 1, not every activity may be represented by an event in the process environment, e.g., execution of transitions *iSP* and *iE* cannot be detected by an event directly. Nevertheless, to keep the paper concise, we require that each decision can be monitored, i.e., transitions that share an input place need to be event monitoring points.

The mechanism above allows obtaining the current state and history of a process, whether it is waiting, which partner it is waiting for, and with which partners it has

interacted, by means of workflow modules and a sparse process log, i.e., detected event monitoring points. However, as one may not want to disclose the detailed behavior of a process to external parties, i.e., the interaction partners, we provide a model that hides internal behavior, yet captures interaction of the process with its partners.

**Definition 4 (Interaction Monitoring Model, Interaction Semantics).** *An interaction monitoring model $I = (V, E)$ is a connected graph of finite sets of interaction activities $V$ and directed edges $E \subseteq V \times V$. The interaction activities provide a partitioning of the transitions of a workflow module $(P, T, F, P_i, P_o)$, i.e., $\pi(T) = \{v_1, v_2, ..., v_n\}$. If transitions of two distinct interaction activities, $v_i$, $v_j$, are connected by a place, these interaction activities are connected by an edge, i.e., $t_r \in v_i \wedge t_s \in v_j \wedge v_i \neq v_j \wedge t_r\bullet \cap \bullet t_s \neq \emptyset \Rightarrow (v_i, v_j) \in E$.*

- *An interaction activity $v \in V$ is waiting, if there is no transition $t \in v$ enabled, but at least one of these transitions is waiting.*
- *An interaction activity $v \in V$ is active, if there is at least one transition $t \in v$ that is enabled or waiting.*

Essentially, an interaction model is an abstraction of the process' workflow module. Different techniques for automatic abstraction have been presented, e.g., [12], but they are not in the scope of this paper. We rather assume that abstraction addresses specific requirements of the use case at hand, i.e., to provide partners with a perspective on the state of the interaction, while hiding sensitive information from them.

In Fig. 2, the partitioning of transitions is represented by three dashed rectangles (i–iii) that correspond to the phases of the process that have been identified, cf. Section 2. The resulting interaction monitoring model is illustrated in Fig. 3, which shows the interaction activities that embrace the corresponding transitions, their involved partners, and their state derived from the marking of the workflow module. A legend is provided in Fig. 4.

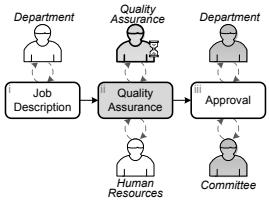


**Fig. 3.** Interaction monitoring model for the job specification harmonization process

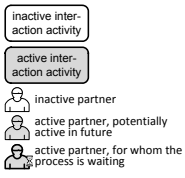


**Fig. 4.** Interaction monitoring model, notation legend

Based on the marking of transitions $t \in v$, we can deduce the state of an interaction activity. If there is at least one transition that is either enabled or waiting, the interaction activity is considered active, as operations may be carried out. If an interaction activity contains only disabled and at least one waiting transitions, then it cannot proceed until the respective messages are received and the waiting transitions fire. Nevertheless, also a waiting activity is active, because in a future point in time, it can proceed.

From the fused workflow modules we know the partner for every transition that is connected with an input or output place, respectively. Hence, for any marking of the net, we can decide, which of the activities are active and which are waiting, and for which partners an activity is waiting.

The interaction activity "Quality Assurance" (ii) is in state waiting, as no transition in the corresponding partition can fire, but transition $!E_{QA}$ is waiting, cf. Fig. 2. Hence, the whole activity waits for the input from Quality Assurance. This is illustrated in the interaction monitoring model by an active (shaded) partner that is annotated with an hourglass. As Human Resources already provided their input and will no further interact with the process in the current interaction activity, the corresponding participant symbol is not shaded, indicating inactivity. Interaction partners of future activities are also shaded as they may be involved as the process advances.

### 3.3 Measuring Interaction Performance

In order to accurately derive the state of each activity of an interaction monitoring model and to compute performance measures on the same level of granularity, a minimal set of event monitoring points is required. This incorporates a monitoring point for each sent message, i.e., $\forall p \in P_o : \bullet p \subseteq T_e$, and one monitoring point for every possibility to enter and leave interaction activities, i.e., for all $t_r \in v_i, t_s \in v_j$ such that $(v_i, v_j) \in E \wedge t_r\bullet \cap \bullet t_s \neq \emptyset$ we require that $t_r \in T_e \vee t_s \in T_e$.

From this information, calculation of various performance measures becomes possible. For instance, the duration of an activity is computed by subtracting from the time of the latest detected event monitoring point the time of the earliest detected monitoring point that belongs to this activity. From active and waiting states of transitions, we can compute the amount of time the activity was actually waiting for a collaboration partner.

Consider, for an example, interaction activity "Job Description" represented by the leftmost dashed rectangle in Fig. 2 and the following log entries, i.e., pairs of event monitoring point and point in time. (Here we refer to the first unlabeled transition as $\tau$). $\{(\tau, 0), (!JD, 3), (!JD_D, 7), (cc, 8), (!JD, 9), (!JD_D, 11), (cc, 13)\}$. After one iteration was required to complete the job description, the result provided from the Department was accepted eventually. In this interaction activity instance, the process was active 13 time units, of which it was waiting $(7 - 3) + (11 - 9) = 6$ time units for the Department.

The overall process duration needs to be computed from the first and the last event monitoring point that have been detected. It is not possible to sum the durations of the interaction activities, as they may be active or waiting concurrently. Finally, computation of statistics, e.g., average or median execution durations for process and interaction activities, is carried out by aggregating durations computed from distinct process instances.

### 3.4 Assumptions and Limitations

Our approach to monitor process interactions is subject to certain assumptions and limitations. First, we assume that, for every partner, we can synthesize a behavior and the

combined behavior. That is, fusing input and output places of the process with output and input places of partners yields a model that can be transformed into a workflow net which is weak sound [9]—a correctness property that ensures the absence of deadlocks, while not all transitions of a workflow module need to be executable.

Further, we require that every process instance strictly follows the model, i.e., event monitoring points are only discovered, when the respective transitions are enabled. Violation of the model could be mitigated by relaxed firing semantics of the model, e.g., if there exists a sound firing sequence that leads to a marking that allows firing an event monitoring point transition, we could accept the event monitoring point. Yet, this is not in the focus of this paper and shall be addressed in future work.

In an interaction monitoring model, edges represent only relaxed ordering semantics of interaction activities. An edge indicates that from one activity another may follow, however, two interaction activities that are connected by an edge may be active concurrently. Also, activities may be skipped. More accurate semantics of these edges could be defined by restricting the abstraction provided through the partitioning of transitions. However, the given abstraction proved effective in practice, where often a coarse-grained differentiation in subsequent process phases is desired.

We resorted to a rather simplistic visual representation of the interaction monitoring model, as the focus of this paper is on the framework to track interaction, detect waiting states, and derive performance evaluations. Much more information can be derived from the workflow modules' structure and the record of event monitoring points, e.g., how often have certain messages been sent, how long did each of these interactions take, and whether interaction activities have been active several times and, if so, how often.

## 4 Implementation

Fig. 5 shows the conceptual architecture of a process monitoring and process performance evaluation system. It comprises an event capturing layer that obtains the event information from various sources. In the aforementioned use case, the events used for monitoring are captured from excel files and business information systems. The event information is correlated with particular process instances, e.g., harmonization of job specification 'clerk for private patients', and a specific activity in the process. Once the event information are correlated to the process instance and its activities and provided to the process monitoring and process performance evaluation components, these can decide which process monitoring points have to be executed, to track the state of the underlying process. Monitoring and evaluation are then provided by means of the mechanisms presented in Section 3.

The health insurance company used the interaction monitoring model to share the state of every process instance with all involved partners. That is, the respective Department can follow the progress of its case, while all involved partners can see, whether the process is stuck due to a missing message. Internally, the OD department uses insights from a more detailed model, i.e., on the same granularity level as the workflow modules, to discover causes for delays and identify cases that violated or are likely to violate the aspired maximum duration of three months to carry out a process instance.

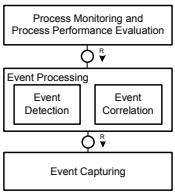


**Fig. 5.** Architecture of a process monitoring and process performance evaluation system

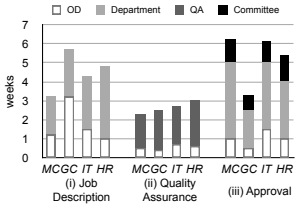


**Fig. 6.** Sample report on time spent in different phases of the process

On the executive level, information of this model is used to evaluate the time spent for harmonizing job specifications and to benchmark the performance of different departments in the context of this process, i.e., to compare response times of the departments and investigate potential bottlenecks at their site. Fig. 6 provides a bar chart that shows how much time the respective departments spent within the various phases. The diagram shows average values of time shares among the four departments *Medical Care* (MC), *Geriatric Care* (GC), IT, and HR, i.e., each bar segment shows how much time each interaction partner contributed to a given phase, whereas the height of a bar shows the length of the phase. Since the phases are subsequent to one another, the average process duration can be computed by summing the durations of the respective phases. From the diagram, we can derive that processes involving the MC and GC department have, in average, met the three months (13 weeks) constraint, whereas cases of the IT and HR department took over 13 weeks in average.

## 5 Related Work

Capabilities to monitor, visualize, and evaluate business process execution are perceived one of the core topics addressed by business process intelligence (BPI) [11], which addresses "managing process execution quality by providing several features, such as analysis, prediction, monitoring, control, and optimization" [6]. Several works discuss the capturing and storing of process execution data for evaluation purposes [6,1,10], but disregard how this information can be used to monitor process interactions.

In [11], the authors argue that process monitoring and analysis are vital to BPI and propose, based on the specific requirements of BPI, a reference architecture, composed of an integration layer, a functional layer, and a visualization layer. The framework presented in this paper targets at the functional and the visualization layer, i.e., provides means to relate event monitoring points with business process state and derive insights. We do not address technical questions, e.g., how actual events are detected in the process environment, and how they are correlated with event monitoring points. A solution toward this is presented in [7].

Dahayanake et al. [4] give an overview of business activity monitoring (BAM) and introduce four classes of BAM systems: pure BAM, discovery-oriented BAM, simulation-oriented BAM, and reporting-oriented BAM. The first is similar to traditional process monitoring, i.e., provides notifications about a certain state, which is already provided

by event monitoring points that found the basis of our approach. Reporting-oriented BAM provides a performance context for running processes and is quite similar to the use case we aim for, where the actual state of a process instance is provided as well as information about the performance evaluation at this stage.

With regards to business process evaluation, the concept of process performance indicators (PPI), the process related form of key performance indicators, is introduced in BPM. Del-Río-Ortega et al. [5] introduce an ontology to define PPIs for measuring process execution performance, such as time, costs, and occurrences. These PPIs can be applied directly on top of our framework, as it provides measurements that can be compared to target values. As these measures can already be provided while the process instance is running, violations of tolerance thresholds can be mitigated before the process instance failed a PPI.

As mentioned earlier, none of the above research addressed monitoring of business process interactions. Rinderle-Ma et al. [13] discuss the need of process views to strengthen the understanding of a business process according to the users' needs. In the same vein, the requirements for a process monitoring system for system-spanning business processes are discussed and evaluated in [3]. The requirements drawn in that work lead the authors to a monitoring framework for visualizing the execution of business processes [2], where interaction diagrams are advocated as one of the most suitable forms to show interaction between several parties participating in a business process execution. However, execution information about partner interaction, such as waiting for a partner's input, are not discussed.

In [14] aspects of so-called federated tasks are discussed, where collaboration across companies is encapsulated in one task, rather than conducted through task interaction. The authors also raise the problem of temporal dependencies of tasks between interacting organizations as well, but owe an answer to the reader. [17] presents an approach how collaboration between partners could be simplified by shifting it to the cloud. In this setting, interaction monitoring capabilities are provided, because workflow engines on both partner sites as well as in the cloud are assumed. However, in our setting neither the organization running the process nor the partners that contribute have a workflow engine in place; a cloud with a workflow engine is not applicable in this setting.

The majority of approaches to process monitoring does not target on monitoring and analysis of interaction with partners during process execution. Especially information about partners that are waiting for a message and according performance evaluations are not addressed so far. The presented framework provides the techniques to detect waiting states and derive information about the responsibilities in a certain interaction activity.

## 6 Conclusion

In this paper we presented a formal framework to capture and express monitoring of business process interactions in non-automated process execution environments. The framework is based on a mechanism to track the state of a process instance even if only few events can be detected and to detect whether certain activities of a running process cannot proceed as they are waiting for a partner. Abstractions of the detailed process

can be used to define the context of performance evaluation and to provide monitoring to collaboration partners.

In a case study from an industry project, we showed how the approach has been applied to derive an interaction monitoring model, and how certain performance evaluations are computed by means of the developed framework. The resulting process interaction monitor increased the transparency among interacting processes of collaboration partners, yet keeping partners independent in their actions, and proved effective in reducing process delays.

In future work, we shall address advanced scenarios that arise from the given limitations and assumptions of our approach. Probably, the most challenging assumption is strict execution compliance, i.e., process executions strictly follow the model. While this has proved effective in the aforementioned use case, it does not generally hold. Different techniques to discover violations of the model's prescription, e.g., an event has been detected although the according monitoring point was not enabled, need to be analyzed.

Based on workflow modules, it is possible to detect, when a process model exhibits certain flaws, e.g., a partner might still wait for a message, while the process has already terminated. By means of model checking, such deficiencies can be mitigated and lead to improved process models that reveal partner interaction.

While we argue that process performance indicators can be applied on top of our framework, cf. Section 5, an implementation is currently missing and shall also be addressed in future work.

## References

1. Azvine, B., Cui, Z., Nauck, D.D., Majeed, B.: Real Time Business Intelligence for the Adaptive Enterprise. In: IEEE CEC/EEE 2006, p. 29 (2006)
2. Bobrik, R.: Konfigurierbare Visualisierung komplexer Prozessmodelle (2008)
3. Bobrik, R., Reichert, M., Bauer, T.: Requirements for the Visualization of System-Spanning Business Processes. In: BPMPM 2005/DEXA 2005, pp. 948–954. IEEE Computer Society Press (2005)
4. Dahanayake, A., Welke, R.J., Cavalheiro, G.: Improving the understanding of BAM technology for real-time decision support. Int. J. Bus. Inf. Syst. 7, 1–26 (2011)
5. del-Río-Ortega, A., Resinas, M., Ruiz-Cortés, A.: Defining Process Performance Indicators: An Ontological Approach. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6426, pp. 555–572. Springer, Heidelberg (2010)
6. Grigori, D., Casati, F., Castellanos, M., Dayal, U., Sayal, M., Shan, M.C.: Business process intelligence. Computers in Industry 53(3), 321–343 (2004)
7. Herzberg, N., Kunze, M., Rogge-Solti, A.: Towards process evaluation in non-automated process execution environments. In: ZEUS 2012, pp. 96–102. CEUR-WS.org. (2012)
8. Lohmann, N., Verbeek, E., Dijkman, R.: Petri Net Transformations for Business Processes – A Survey. In: Jensen, K., van der Aalst, W.M.P. (eds.) ToPNoC II. LNCS, vol. 5460, pp. 46–63. Springer, Heidelberg (2009)
9. Martens, A.: On Compatibility of Web Services. Petri Net Newsletter 65, 12–20 (2003)
10. Melchert, F., Winter, R., Klesse, M., Romano Jr., N.C.: Aligning process automation and business intelligence to support corporate performance management. In: AMCIS 2004, pp. 4053–4063. Association for Information Systems (2004)
11. Mutschler, B., Reichert, M.U.: Aktuelles schlagwort: Business process intelligence. EMISA Forum 26(1), 27–31 (2006)

12. Polyvyanyy, A., Smirnov, S., Weske, M.: The Triconnected Abstraction of Process Models. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 229–244. Springer, Heidelberg (2009)
13. Rinderle, S., Bobrik, R., Reichert, M., Bauer, T.: Business Process Vizualization - Use Cases, Challenges, Solutions, pp. 204–211 (2004)
14. Unger, T., Wagner, S.: Collaboration Aspects of Human Tasks. In: zur Muehlen, M., Su, J. (eds.) BPM 2010 Workshops. LNBIP, vol. 66, pp. 579–590. Springer, Heidelberg (2011)
15. van der Aalst, W., van Hee, K.: Workflow Management: Models, Methods, and Systems. MIT Press, Cambridge (2004)
16. van der Aalst, W.: The Application of Petri Nets to Workflow Management. Journal of Circuits, Systems, and Computers 8(1), 21–66 (1998)
17. Wagner, S., Kopp, O., Leymann, F.: Towards Choreography-based Process Distribution in the Cloud. In: CCIS 2011, pp. 490–494. IEEE Xplore (2011)
18. Wolf, K.: Does My Service Have Partners? In: Jensen, K., van der Aalst, W.M.P. (eds.) ToPNoC II. LNCS, vol. 5460, pp. 152–171. Springer, Heidelberg (2009)