Towards Process Evaluation in Non-automated Process Execution Environments*

Nico Herzberg, Matthias Kunze, Andreas Rogge-Solti

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

Abstract. Process models gained more and more significance to carry out an organization's operations. Besides documentation purposes, organizations strive to evaluate their executed processes in terms of performance and conformance. However, this is far from trivial: As most processes are still carried out manually, only few effects can be tracked and are typically not related to process instances. In this paper, we propose an architecture that defines event monitoring points: Elementary state transitions of a process instance that are bound to a configuration to discover events from a process agnostic technical environment. We discuss applications of this architecture, towards monitoring, performance measurement, and execution conformance.

1 Introduction

Central to managing an organization in a process-oriented fashion are process models, as they explicitly capture the operations carried out and are used, among others, for documentation, certification, and enactment. There is a large body of work that addresses automatic orchestration of business processes through process-aware information systems (PAIS), while the majority of processes are still carried out manually.

In the latter case, it is difficult to track the execution of a process for different reasons: (a) Tasks cannot be tracked, if they have no observable side effect in IT systems, e.g., examining a patient on a ward round, and thus would need to be recorded by hand, which is time consuming and prone to errors. (b) In the absence of a PAIS, there is no central system to collect information about process advancements and thus, (c) valuable information about progress is contained in various systems, such as ERP or CRM systems, but cannot easily be related to a process model. As a result, process execution information is scattered among the IT landscape and only few events of a process can be captured at all. Reconstruction of these events requires explicitly defined methods to access systems, combine relevant information, and correlate it with a process instance. Approaches towards monitoring, performance measurement, and conformance verification assume the presence of a complete event log, which is not the case according to the above discussion. Hence, they cannot be properly applied.

* This work is supported and funded by the German Federal Ministry of Education and Research (01IS10039B)

In an ongoing research project, PIGE[1]—Process Intelligence in Health Care—we encountered above obstacles in the University Hospital of Jena, where the performance of clinical pathways, i.e., disease treatment processes, shall be measured and evaluated against key performance indicators.

In this paper, we present a basic architecture that explicitly addresses these issues, and propose a solution, where so-called event monitoring points are defined at particular points of a process. An event monitoring point is bound to a certain state transition in the process and contains information, how this event can be discovered in a heterogeneous IT landscape. We further discuss, how this can be used to monitor process instances, apply key performance indicators for performance measurement and examine running processes for their conformance to given models.

2 Architecture

As our approach towards process evaluation is tightly coupled with process models and state transitions, we first introduce these concepts and then illustrate how this manifests in a system overview.

2.1 Fundamentals

Our definition of a process model subsumes a connected graph consisting of nodes N and edges F. This covers commonly used process modeling languages, such as BPMN [8] and EPC [4].

Definition 1 (Process Model). *A process model is a tuple $P = (N, F)$, where N is the set of control flow nodes and $F \subseteq N \times N$ is the flow relation that captures ordering constraints of the process execution.*

Note, that other modeling notations, such as value chains, where N represents coarse grained units of work and F represents the execution order of these work units is also covered by this generic definition. In BPMN, N is partitioned into *activities*, *gateways*, and *events*, whereas F represents sequence flow among these nodes, for an example refer to Fig. 1. For each of these node types, we envision a state-based life cycle model, where we reserve the flexibility, to assign a unique life cycle model to any node. Life cycles of process nodes have been exhaustively discussed in literature [11,9]. Thus, we employ a generic life cycle model.

Definition 2 (Life Cycle Model). *A life cycle model $L = (\Sigma, S, T)$ consists of an event alphabet Σ, states S and state transitions T. $\mathcal{L}$ is the universe of life cycle models.*

Let $P = (N, F)$ be a process model. There exists a function $lc : N \to \mathcal{L}$ that assigns a life cycle model to every node $n \in N$ of P.

State transitions are the most elementary facts that can be leveraged to monitor progress during process enactment. The set of all state transitions of a process model is comprised by $\bigcup_{n \in N} \{(n, t) | t \in T_{lc(n)}\}$, each of which could be potentially captured. However, as

[1] `http://pige-projekt.de`

explained in Section 1, only a subset of those can or shall actually be monitored. We refer to these state transitions of interest as event monitoring points.

Definition 3 (Event Class, Event Monitoring Point). *C is a set of event classes indicating the nature of an event. Let $P = (N, F)$ be a process model. An event monitoring point is a tuple $M = (n, t, c)$, where $n \in N$ is a node, $t \in T_{lc(n)}$ is a state transition within the life cycle of node n, and $c \in C$ is the event class to be monitored.*

Event monitoring points of a process model are selected state transitions, for which information can be retrieved from the process environment and they can be bound to an implementation. Event classes are used to specify the measurement an event monitoring point shall provide, e.g., time, counters, or cost.

Definition 4 (Binding, Implementation). *Let $\mathcal{M}$ be the set of defined event monitoring points of a process model P. A binding is a function $bind : \mathcal{M} \rightarrow \mathcal{I}$ assigning an implementation to an event monitoring point, where $\mathcal{I}$ is the universe of implementations, i.e., rules and methods to capture an event in the process execution environment.*

This definition allows to implement event monitoring points in several ways, e.g., as a database query, as a service request, a calculation method, as a stream processing filter, or reading a log entry. An important aspect of the binding is correlation, i.e., identification of process instances and events that refer to this process instance. As we assume no central process orchestration control, an implementation needs to account for correlation by combining data retrieved from accessed systems.

2.2 System Overview

Our example illustrates a sample business process that describes the admission and examination of a patient, who needs a liver transplant. If the patient is eligible for transplantation, she is enlisted to a European-wide register, Eurotransplant.

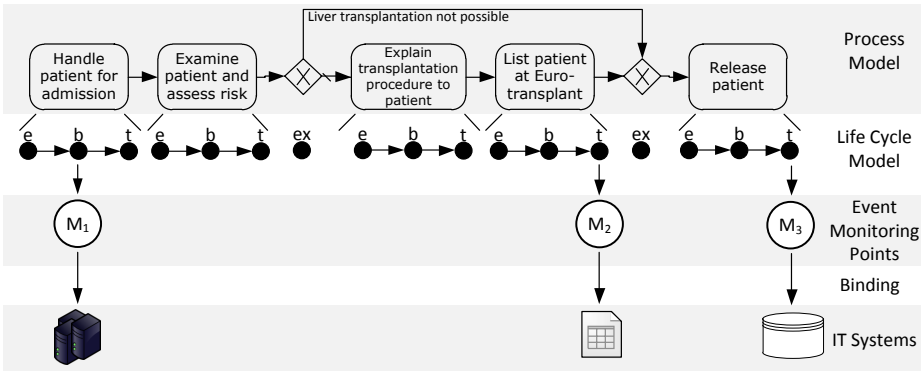

Fig. 1. Infrastructure: Example Process with life cycles ⟨**e**nable, **b**egin, **t**erminate⟩ for activities and ⟨**ex**ecute⟩ for gateways, selected event monitoring points, and bindings to process environment.

As explained in Section 1, we assume a process model that is the basis for process execution, which takes place in a certain environment. Part of this environment are IT systems that do not necessarily log events of process enactment, but capture the effects of activities nonetheless. These data are produced as a side effect while using IT systems to record information and are very valuable to describe the progress of a process execution. In the example, the information about the patient admission is stored in a hospital information system, the information about the confirmation of Eurotransplant listing is stored in a spread sheet file, and the patient release information is stored in a separate database.

Based on the available information, state-transitions of interest are chosen and defined as event monitoring points with the corresponding event class. In the example in Fig. 1, we keep matters simple and consider only event monitoring points of class *time*. Nevertheless, it is possible to have more than one event monitoring point defined for a certain state-transition, when it is necessary to monitor multiple event classes. In the example, we identified three event monitoring points:

- M_1 is the state-transition *begin* of activity *Handle patient for admission*
- M_2 is the state transition *terminate* of activity *List patient at Eurotransplant*
- M_3 is the state-transition *terminate* of activity *Release patient*

The other activities do not have observable side effects in IT systems, and hence cannot be tracked.

Once the event monitoring points are defined, they can be bound to an implementation. In the example shown in Fig. 1 the implementation of

- M_1 is the call of a web service provided by the hospital information system,
- M_2 is the content of a certain cell in a spread sheet file, and
- M_3 is an SQL query to retrieve data from a relational database.

In the given context, correlation of events to process instances is possible through a unique treatment case id that is stored consistently across all IT systems and can be matched to a patient.

This implementation will be used during the enactment of any instance of the process to access the IT systems and discover the occurrence of events defined in event monitoring points. Once events are extracted and correlated with process instances, they can be used to visualize the data in a monitoring user interface or to set the stage for measurements and KPIs. The causal dependencies between event monitoring points are defined by the process model and the life cycle model of its nodes. This can be leveraged for conformance checking and notification, if a deviation from the process model is detected.

During runtime, the monitoring system based on the discussed architecture is querying the process execution data, resp. event data, online from the IT systems. The method of querying the data is encapsulated in the implementations that are bound to the event monitoring points. There is no notification from the IT systems shown in Fig. 1. Thus, the architecture is following a pull approach, not a push mechanism. Pulling is necessary because in this real-world scenario the IT landscape is not enabled for pushing messages/events most times to interested listeners. In addition the running systems should not be extended by introducing a process monitoring system. Nevertheless, if the IT landscape supports pushing events to the monitoring platform, that would be the

preferable way to reduce bandwidth usage in the network. This implementation detail is not affected by the proposed architecture.

While the proposed architecture and its application are rather generic, we resorted to a simple process consisting of activities and alternative (XOR) gateways, each having brief life cycle models, in the example in Fig. 1. In practice, the architecture could be easily extended to data objects, control and data flow relations, for instance, if a transition from one activity to another would explicitly be manifested in a state change of a central data artifact. Also, every single node could have its own unique life cycle model.

2.3 Case Study

In the hospital setting that we encounter in our research project PIGE, we face the issue that the records about events during a treatment are distributed over several IT systems. Depending on the available systems, some of the steps during treatment are logged in a spread sheet file, some in a SAP Healthcare system and others in separate databases. The University Hospital of Jena elicited detailed process models for the clinical pathways of the liver transplant surgery and the colorectal carcinoma, along with milestones that resemble event monitoring points. The goal of the project is to provide process intelligence and enrich the process models with runtime information about the treatment cases. Process intelligence is enabled by answering analytical questions such as:

- How long does it take from the initial contact with the patient until evaluation for the liver transplant is finished?
- How many emergency patients were treated?
- Which treatment methods were applied?

First, we wanted to enable *monitoring* [5] of a process. One major application of monitoring is the definition of target performance values, which are *key performance indicators* (KPIs) if they are relevant for success, in the model. This allows the detection of deviations from planned time and cost limits in a process. The monitoring system can raise alerts and reminders to inform the responsible process owners and the process participants about delays or exceeding costs. In addition, there is a huge gain of *transparency*, as it becomes visible at which stage a current process instance is in a process model. Note that, while process models can be quite detailed, in the given setting only few event monitoring points can be defined. Thus, there exist unmonitored blank spots in the process model, where KPIs cannot be attached to. Second, the *prediction* of time and cost of an instance becomes much more accurate, when real-time execution information of a process is available and bound to a model. It can be used for improving efficiency by planning resources more accurately. Third, *conformance checking* helps to detect deviations, e.g., missing necessary steps, or the absence of recording them in specified IT systems. Reacting to deviations is very important, as reminders for drug administration and other treatment steps are beneficial to increase quality of care.

In the PIGE research project, we want to assess existing process evaluation methods and tailor them for this specific setting, where execution information for only few activities in the process model is available.

3 Related Work

One problem that has to be addressed when different event sources for a process exist is correlation of events to one process instance. Motahari-Nezhad et al. [7] provide algorithms to determine correlation sets on different attributes of events for distributed environments. They use methods of atomic, conjunctive, and disjunctive correlation conditions and heuristics to find correlating groups. The aspects of correlation are also relevant for this paper, while the focus is on how to map correlated information from different sources to a process model in a flexible architecture.

Process mining [10] is a discipline that can be used on top of correlated information merged in an *event log* to extract all kinds of process information, e.g., process models generated from real-life event data, execution times and conformance checks to existing models. The main difference to the architecture presented in this paper is that we utilize a top-down approach of connecting (detailed) process models to process information, while process mining is a bottom-up approach based on logs.

Closely related to process monitoring is the topic of process performance measurement, or business process intelligence, that addresses "managing process execution quality by providing several features, such as analysis, prediction, monitoring, control, and optimization" [3]. There is a considerable body of work that addresses means to capture and store process execution data and offer it for evaluation purposes [3,6,1]. Del-Río-Ortega et al. [2] present a comprehensive ontology to define process performance indicators that measure execution time, occurrence, and costs of processes. However, the majority of such approaches rely on a complete log, i.e., a protocol of every state transition of a process instance. In contrast, the architecture we presented lays the ground work for these approaches in the absence of a complete log.

4 Discussion and Future Work

This paper shows a general and flexible architecture to monitoring and performance evaluation for non-automated process execution environments. In practice, manually executed processes are common, because automation is not always profitable or feasible depending on the process and domain. However, some process information is often available in IT systems and can be exploited for process monitoring. With the described architecture it is possible to define event monitoring points in a process model and bind them to respective implementations. During runtime, these implementations are used to pull events from the IT systems updating the process model for monitoring purposes.

This setting raises additional questions for future work that includes dealing with observed deviations from the modeled process, e.g., missing events, duplicate events, or violation of ordering constraints imposed by the process model.

A complex model of many activities that only has few event monitoring points may not be well suited to display the state of a process instance, as it lacks information for most of the activities. A better representation would merge fragments of a process model, based on the available monitoring data. The result would be an abstraction, where a node represents a precisely measurable entity of work. On the other hand, it may be useful to abstract the process into a very coarse grained representation, e.g., to communicate it to

external stakeholders, which in turn requires aggregating and adjusting event monitoring points.

In a separate stream of work, we aim at implementation strategies for this approach. Currently, the binding of an event monitoring point is laborious work on the edge between business requirements, e.g., KPI definitions, and technical capabilities. Therefore, we envision decoupling the work of process experts and implementers by means of a service-based bindings. An additional layer decouples the IT systems from event monitoring points and can provide a flexible event distribution model, e.g. publish-subscribe, or caching. These services can be used by the by process modelers to configure event monitoring points. An adequate language for event monitoring point configuration is required to ease the currently laborious implementation process.

References

1. B. Azvine, Z. Cui, DD Nauck, and B. Majeed. Real time business intelligence for the adaptive enterprise. In *IEEE-CEC, 2006.*, pages 29–29. IEEE, 2006.
2. A. Del-Río-Ortega, M. Resinas, and A. Ruiz-Cortés. Defining process performance indicators: an ontological approach. *OTM 2010*, pages 555–572, 2010.
3. D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, and M.C. Shan. Business process intelligence. *Computers in Industry*, 53(3):321–343, April 2004.
4. G. Keller, M. Nüttgens, and A.W. Scheer. Semantische Prozessmodellierung auf der Grundlage "ereignisgesteuerter Prozessketten (epk)". *Veröffentlichungen des Instituts für Wirtschaftsinformatik*, 89, 1992.
5. D.W. McCoy. Business activity monitoring: Calm before the storm. *Gartner Research, ID: LE-15-9727*, 2002.
6. F. Melchert, R. Winter, M. Klesse, and N.C.Jr. Romano. Aligning process automation and business intelligence to support corporate performance management. In *AMCIS'2004*, pages 4053–4063, New York, 2004. Association for Information Systems.
7. H.R. Motahari-Nezhad, R. Saint-Paul, F. Casati, and B. Benatallah. Event correlation for process discovery from web service interaction logs. *VLDB Journal*, 20(3):417–444, 2011.
8. OMG. Business Process Model and Notation (BPMN) 2.0 Specification, January 2011. `http://www.omg.org/spec/BPMN/2.0/PDF`.
9. N. Russell, W.M.P. van der Aalst, A.H.M. ter Hofstede, and D. Edmond. Workflow resource patterns: Identification, representation and tool support. In *Advanced Information Systems Engineering*, pages 216–232. Springer, 2005.
10. W.M.P. Van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer-Verlag New York Inc, 2011.
11. M. Weske. *Business process management: concepts, languages, architectures*. Springer-Verlag New York Inc, 2007.