Proactive Decision Support During Business Process Execution

Kimon Batoulis
Hasso Plattner Institute at the University of Potsdam
Prof.-Dr.-Helmert-Str. 2–3, D-14482 Potsdam, Germany
Kimon.Batoulis@student.hpi.uni-potsdam.de

Abstract: The execution of business processes produces lots of events that can be used by complex event processing systems to analyze and improve the processes. Typically, events are stored in an event log repository that can be used to identify meaningful patterns of events, such that it is possible to react to them during process execution. However, in many cases it is beneficial to deal with those events before they actually occur to avoid an undesirable outcome, e.g., machine failure or poor performance indicator. Therefore, we present a system architecture connecting the operation of a process engine with a proactive framework. The framework forecasts events, provides the best corresponding action and generates appropriate business rules that can be used by the process engine to make optimal decisions during process runtime. An elaborated example demonstrates the utility of our concept.

Keywords: complex event processing, proactive computing, operational support, business process management

1 Introduction

By introducting the term complex event processing (CEP) [Luc01], Luckham emphasized the need for methods and tools supporting the management of today's IT systems being characterized by their event-driven nature. This is apparent by the large amount of scientific work that has been published on the subject since. In principle, CEP is about deriving complex events on the basis of patterns of events and reacting to them appropriately, e.g., by executing simple actions like buying a certain amount of stocks [EB09].

CEP can be related to business process management (BPM). BPM is concerned with concepts and techniques that can be used to model, enact and analyze business processes [Wes12]. The connection to CEP is established by realizing that the execution of business processes produces events that can be consumed by CEP systems. Those events typically have a meaning in the context of the business process or influence what will happen at a later point in time. CEP techniques can be used to respond to the occurring events in a reactive manner. However, in [EN10] it has been noted that CEP could not only be used in a reactive but also

A. Baumgraß, N. Herzberg, G. Kappel, J. Mendling, A. Meyer, S. Rinderle-Ma (eds.): Proceedings on Inter-Organizational Process Modeling and Event Processing in Business Process Management, Vienna, Austria, 12-06-2014, published at http://ceur-ws.org

in a proactive manner. The difference is that in proactive event-driven systems prediction techniques are applied in order to forecast future events and react to them *before* they occur, instead of after [EE11, EEF12, WGET12]. A little over a decade ago this has been considered "beyond the state of the art" [Luc01, p. 46], but in recent years the research field of proactive computing in CEP has evolved [EE11, EEF12, WGET12]. not only in CEP [EE11, EEF12, WGET12] but also in different areas of computer science, e.g., in dependable systems [SLM10]. The advantages are that it is possible to prevent the event from occurring, e.g., in case of failure of a machine, or to influence it in some way such that it leads to an improved performance indicator value. A detailed example dealing with the latter case is given in Section 2.

The remainder of this paper is structured as follows. Section 2 describes the example used to illustrate our concept and formulates the requirements for proactive decision support in a generic way. Section 3 explains how this problem can be approached by a proactive framework. Furthermore, it describes the complete system architecture of our concept and shows how the running example can be applied to it. This is followed by related work in Section 4. Section 5 concludes the paper.

2 Motivating Example

As an example of how the application of proactive techniques can improve a performance indicator value like time to repair, we present an example elicited and adapted from an event log of a simulated process used in the tutorial of the process mining tool ProM [1] and illustrated as a BPMN diagram in Figure 1.

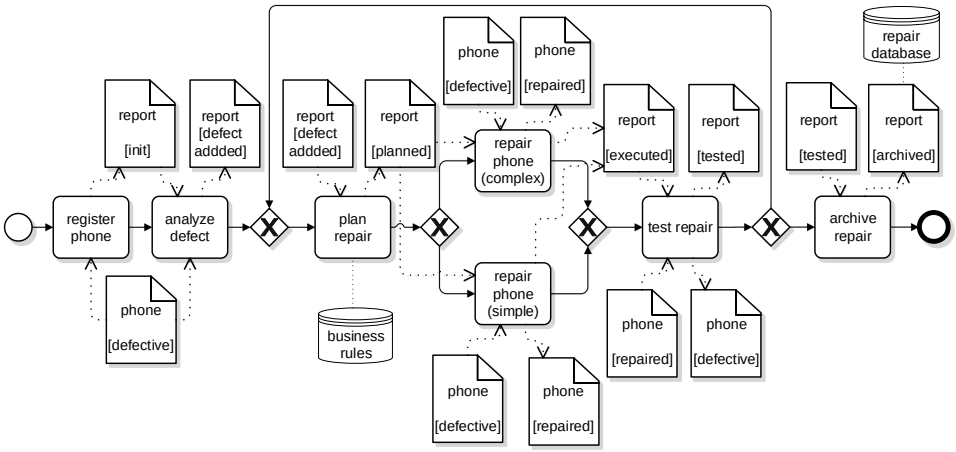

Figure 1: Telephone repair process

[1] http://www.promtools.org/prom6/

In this scenario a telephone company takes repair requests of its customers. There are two data objects in use: the defective phone and a repair report that holds various kinds of information which is updated during the execution of the process. The report's data model is shown in Figure 2. After the phone has been registered, it is analyzed to determine the type of the defect and the report is updated appropriately. Subsequently, the repair procedure is planned in that values for the fields *repairType*, *solver* and *tester* are set. The value of *repairType* is dependent on the defect type (some defects are simple, others are complex) and is the basis for the following gateway decision, leading to a simple or complex repair. The *solver* field determines the technician assigned to the current repair case and *tester* indicates who will test the phone afterwards. The activity *plan repair* is connected to a data storage containing business rules. Those rules are used to set the values of the report and will be mentioned again below. If the repair was successful, it can be archived, otherwise it needs to be replanned and restarted.

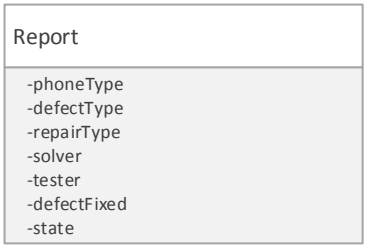

Figure 2: Repair report data model

In our example, the telephone company has three kinds of customers, namely gold, silver and standard. Different kinds of customers have different service-level agreements (SLA). For gold customers it is stated as follows: *For gold customers the repair time is faster than the mean repair time in 2/3 of the cases.* For silver customers the same holds in 1/2 of the cases and for standard customers no guarantees exist.

Thus, especially when dealing with gold customer phones, the objective is to minimize the time to repair in order to conform to the SLA. We could pursue this intent by first recognizing that, on average, the activity *repair (complex)* takes longer than *repair (simple)*. Hence, preferring the latter might be beneficial. However, the simple repair may not really solve the problem with the phone so that the test activity afterwards reveals that the repair procedure needs to be executed once again, thereby increasing the time to repair. So we are better off sticking to the result of *analyze defect*. Yet, analyzing further, we realize that not only the type of defect is crucial for deciding whether a simple or a complex repair should be executed but also other context information like the experience of available technicians. For instance, consider a scenario in which the detected defect could be solved with *repair (simple)* and that it has a certain probability of failing, meaning that the repair needs to be executed again with that probability. However, suppose that at the current time there are very experienced technicians available that can quickly execute a complex repair with a much lower probability of failing the test. Thus, we would like to recommend this action in case of gold customers.

The example given above demonstrates that there are two aspects that need to be covered. On the one hand, a prediction framework is necessary that takes as inputs the events that occurred in the system like a particular result of the *analyze defect* activity and that outputs probabilistic events. In the described scenario, the

output could be the probabilistic event that the process will be faster than the mean process time.

On the other hand, the probabilistic events need to be processed by an act framework that implements the proactive behavior by choosing actions that provide the highest utility. In the example above the utility of executing the action *repair (simple)* is weighed against that of *repair (complex)*. The decision can be based on several factors such as available resources (experienced technicians), process data (result of *analyze defect*), overall execution times and costs (more experienced technicians are more expensive).

3 Proactive Framework and System Architecture

In [EE11] a suggestion is made how to implement these two aspects of the overall framework. For example, to predict the occurrence of an event with some probability, a Bayesian network (BN) [Pea88] could be constructed (automatically). Such a network has a qualitative and a quantitative aspect. The qualitative component is a directed graph whose nodes represent random variables and whose edges represent their dependencies. The quantitative component then assigns probabilities to those dependencies. With a BN, one could state that the probability of failing the test depends on the repair type and technician chosen earlier and how this probability affects the objective of being faster than the mean repair time in 2/3 of the cases.

For the act framework some optimization procedure is necessary. Here, Markov decision processes (MDPs) have been proposed [EE11] to model a sequence of decision points over time. However, MDPs concentrate on unstructured problems involving very long or even infinite sequences of decisions [KF09]. Consequently, they are not suited to be applied to business processes characterized by their structured execution with well defined start and end events. Rather, we propose the use of influence diagrams, which essentially are BNs enhanced with decision and utility nodes. Thus, one can compute how decisions like executing a simple repair influence the probabilities of other random variables. Furthermore, one can define the utility of each combination of values of the random variables and thus compute the action/decision that yields the highest utility value.

In this work, we connect the proactive framework with the operation of a process engine such that it is possible to incorporate the proactive behavior into running business processes. This is achieved by realizing that the actions specified by the influence diagram can be stated as business rules [OMG13] that can be used to guide the process engine during decision making. In the running example, business rules are stored in the business rules storage associated with the *plan repair* activity in Figure 1. Depending on the kind of customer (gold, silver, standard) and thus the concrete SLA, the business rules will prescribe how the values of the repair report are to be set. For example, in case of gold customers the rule could say that a complex repair should be executed by a technician known to have a high success rate, such that the SLA is not violated.

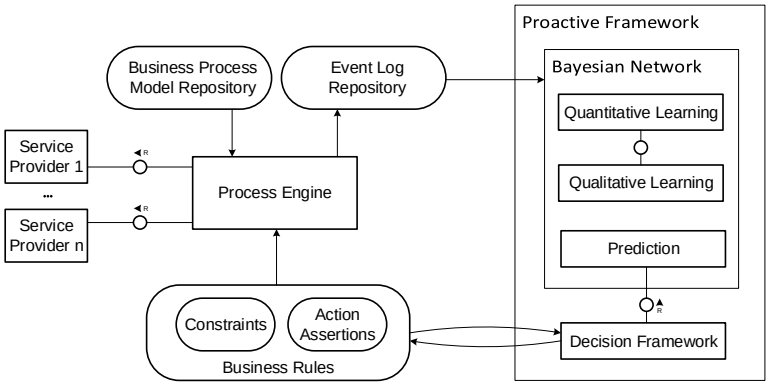

Figure 3: System architecture

The complete system architecture is illustrated in Figure 3. Business processes such as the phone repair process are executed by a process engine according to their models. This generates events such as the determined defect type that are stored in an appropriate repository. The events are then used by the BN component of the proactive framework to model the qualitative and quantitative dependencies between the events (interpreted as random variables). For instance, a dependency between the phone type and the defect type could be established and also how often a certain phone has a specific defect.

The novelty of our concept rests on the fact that the decision or optimization component interacts with a business rules storage. On the one hand, the component reads constraints from the storage, which in the example above would be the SLAs. On the basis of those constraints it makes requests to the BN in order to find action assertions that fulfill the constraints with the predicted probability. Concerning the repair process, this component could try different combinations of repair types and solvers and ask for the probability of being faster than the mean process time, given those combinations.

On the other hand, the assertions in turn are written to the business rules storage and can be read by the process engine during execution. In the example in Figure 1, this corresponds to the *plan repair* activity reading from the business rules storage. An example rule would be to instruct the experienced solver *Solver3* to execute a complex repair, given a gold customer phone of type $T3$ and defect type 5.

Note that the event log repository in Figure 3 is updated whenever a process instance is executed. Consequently, the proactive framework has more information available and can construct more accurate models, which in turn lead to more accurate actions assertions to be used in future process executions.

4 Related Work

In the area of CEP, important publications dealing with proactive behavior include [EE11] and [EEF12], where the current conceptual architecture of CEP presented in [EN10] is extended to accommodate new artifacts, like predictive and proactive agents in addition to standard event processing agents. Furthermore, concepts for implementing these new artifacts are investigated, like BNs for predictions and MDPs for actions.

There also exists a lot of work dealing only with the uncertainty present in event-driven and rule-based systems. A good overview of the nature of uncertainty in those systems is given in [WGE06]. This treatment of uncertainty is extended in [WGET08, WGET12], where the problem of the impact of uncertainty at the event source on the materialization uncertainty at the event sink is tackled with the help of probabilistic reasoning using dynamic BNs. However, this work assumes that rules for event generation and the associated probabilities are already known. This is often not the case, because even for domain experts it is hard to accurately specify those rules.

5 Conclusion

This paper presented a system architecture that couples the operation of a process engine with a proactive framework. The latter consists of a prediction and an action component. Thus, the probability of occurrence of an event can be computed on the basis of an event log and correponding actions are proposed that deal with those events. The actions depend on the specific context of the current process instance and are supplied to the process engine as business rules that are continuously updated as more information is gathered.

In future work, more use cases will demonstrate the utility of our concept. Also, BN learning algorithms tailored to the domain of BPM and CEP will be investigated and the concept of generating business rules from an optimization component like an influence diagram that can be used by a process engine needs to be detailed.

References

[EB09] Michael Eckert and François Bry. Complex Event Processing (CEP). *Informatik-Spektrum*, 32(2), 2009.

[EE11] Yagil Engel and Opher Etzion. Towards Proactive Event-driven Computing. DEBS '11, New York, 2011. ACM.

[EEF12] Yagil Engel, Opher Etzion, and Zohar Feldman. A Basic Model for Proactive Event-driven Computing. DEBS '12, New York, 2012. ACM.

[EN10] Opher Etzion and Peter Niblett. *Event Processing in Action*. Manning Publications Co., Greenwich, 1st edition, 2010.

[KF09] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[Luc01] David C. Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Longman Publishing Co., Inc., 2001.

[OMG13] OMG. Semantics Of Business Vocabulary And Business Rules (SBVR), V1.2, 2013.

[Pea88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, 1988.

[SLM10] Felix Salfner, Maren Lenk, and Miroslaw Malek. A Survey of Online Failure Prediction Methods. *ACM Comput. Surv.*, 42(3):10:1–10:42, March 2010.

[Wes12] Mathias Weske. *Business Process Management - Concepts, Languages, Architectures, 2nd Edition*. Springer, 2012.

[WGE06] Segev Wasserkrug, Avigdor Gal, and Opher Etzion. A Taxonomy and Representation of Sources of Uncertainty in Active Systems. NGITS'06, Berlin, Heidelberg, 2006. Springer-Verlag.

[WGET08] Segev Wasserkrug, Avigdor Gal, Opher Etzion, and Yulia Turchin. Complex Event Processing over Uncertain Data. DEBS '08, New York, 2008. ACM.

[WGET12] Segev Wasserkrug, Avigdor Gal, Opher Etzion, and Yulia Turchin. Efficient Processing of Uncertain Events in Rule-Based Systems. *IEEE Trans. Knowl. Data Eng.*, 24(1), 2012.