Prescriptive Analytics for Recommendation-Based Business Process Optimization

Christoph Gröger, Holger Schwarz, and Bernhard Mitschang

Institute of Parallel and Distributed Systems
University of Stuttgart Universitätsstr. 38, 70569 Stuttgart, Germany
{christoph.groeger,holger.schwarz,
bernhard.mitschang}@ipvs.uni-stuttgart.de

Abstract. Continuously improved business processes are a central success factor for companies. Yet, existing data analytics do not fully exploit the data generated during process execution. Particularly, they miss prescriptive techniques to transform analysis results into improvement actions. In this paper, we present the data-mining-driven concept of recommendation-based business process optimization on top of a holistic process warehouse. It prescriptively generates action recommendations during process execution to avoid a predicted metric deviation. We discuss data mining techniques and data structures for real-time prediction and recommendation generation and present a proof of concept based on a prototypical implementation in manufacturing.

Keywords: Prescriptive Analytics, Process Optimization, Process Warehouse, Data Mining, Business Intelligence, Decision Support.

1 Introduction

Today, adaptive and continuously improved business processes play a key role for companies to stay competitive. At this, the digitalization of process execution activities as well as the increasing use of sensor technologies lead to enormous amounts of data, from workflow execution data and machine data to quality data, posing a great potential for analytics-driven process improvement [1, 2].

Yet, existing process analytics in industry practice, e.g., as part of business activity monitoring approaches [3], do not fully exploit the valuable knowledge hidden in these huge amounts of data due to the following limitations: (1) they do not make use of prescriptive techniques to transform analysis results into concrete improvement actions leaving this step completely up to the subjective judgment of the user; (2) they do not integrate process data and operational data, e.g., from workflow management systems and enterprise resource planning systems, to take a holistic view on all process aspects; (3) the actual optimization is conducted ex-post after the completion of the process in contrast to a proactive improvement during process execution.

To address these issues, we present the data-mining-driven concept of *recommendation-based business process optimization (rBPO)* supporting adaptive

W. Abramowicz and A. Kokkinaki (Eds.): BIS 2014, LNBIP 176, pp. 25–37, 2014.

and continuously optimized business processes (see Fig. 1). rBPO exploits prescriptive analytics and proactively generates action recommendations during process execution in order to avoid a predicted metric deviation. It is based on a holistic process warehouse and employs classification techniques for real-time prediction and recommendation generation. For example, a worker is warned during process execution that the entire process is likely to run out of time, even if the current lead time meets the requirements. Then, a corresponding hint, e.g., to adjust a resource setting, is generated using data on past process executions in order to speed up processing and avoid the metric overrun. Thus, rBPO focuses on data-driven process optimization at runtime, not on classical process model improvement during design-time or ex-post analysis.

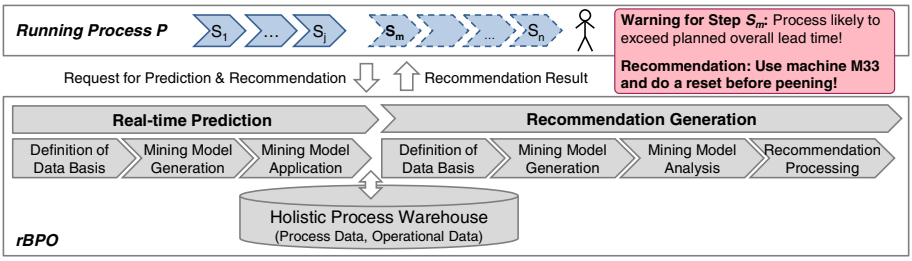

Fig. 1. Recommendation-based Business Process Optimization (rBPO)

The remainder of this paper is organized as follows: In Sec. 2, we define general requirements and present the basic approach of rBPO. Its two major components, real-time prediction and recommendation generation, are detailed in Sec. 3 and 4. Our proof of concept based on a prototypical implementation in the manufacturing industry is described in Sec. 5. Related work and a comparative evaluation of rBPO are discussed in Sec. 6. Finally, we conclude in Sec. 7 and highlight future work.

2 Requirements and Basic Approach

From a process management perspective, metrics are the basis for process optimization. The earlier potential metric deviations, e.g., excessive lead times, are detected during process execution, the more likely they can be avoided [4]. On this basis, we define the following core requirements (R_i) for rBPO.

The approach has to support a *metrics-based goal definition (R1)* and facilitate metric prediction and recommendation generation *proactively during process execution (R2)*. It should make use of all data generated across the entire business process in a holistic data basis. That is, it has to *integrate process data and operational data (R3)*. Process data is flow-oriented and comprises process execution data, i.e., process events, and process model data. Operational data is subject-oriented and provides additional information on process subjects like employees or machines [5]. Thereby, *recommendation generation should be adaptive (R4)* by exploiting the continuously growing data on completed process executions. Besides, recommendations should

comprise multiple actions (R5) in order to achieve the goal, e.g., by recommending both the specific resource to use and the corresponding resource settings.

To realize these requirements, rBPO comprises two major components, namely *real-time prediction* and *recommendation generation*, on top of a *holistic process warehouse (PWH)* (see Fig. 1). The PWH integrates process data and operational data across the entire process and additionally stores analysis results, e.g., data mining models, to enable their reuse.

It has to be remarked that rBPO is a universal concept which can be applied to different process domains like workflow-based processes or manufacturing processes as long as there exists a suitable PWH. In this paper, we focus on manufacturing processes because we developed a holistic process warehouse for manufacturing in our previous work [6]. As a running example throughout the paper, we refer to the manufacturing of steel springs in the automotive industry as described in [7]. It comprises amongst others steps for winding, tempering and shot peening of springs. Besides, an approach towards a holistic process warehouse for workflows can be found in [5] and may be used for rBPO, too.

To keep our approach generic, we only assume that a process P consists of n steps S_i with $P = \{S_1, \ldots, S_n\} \wedge 1 \leq i \leq n$. These steps may be executed not only in sequential but also in parallel and branched structures. Generally, rBPO can be applied to arbitrary process structures. For the sake of understandability, we refer to a sequential process in our examples where step S_{i+1} is executed directly after S_i.

Table 1. Data basis with process instance data including the running instance *i400*

Process Instance ID	Step1 Machine ID	Step1 Winding Speed	Step1 Empl ID	Step1 Empl Qualific	Step2 Machine ID	Step2 Tempering Temperature	Step3 Machine ID	Step3 Peening Reset	Step3 Peening Duration	...	Metric
i100	M12	120	E331	5	M23	290	M33	1	23		OK
i200	M12	135	E332	2	M23	291	M33	0	28		NotOK
i300	M12	135	E321	1	M23	290	M34	0	29		NotOK
i400	M12	121	E321	1	M23	285					
...			...	...							...

The *starting point of rBPO* is a holistic data basis with instance data about a process provided by the PWH in a denormalized structure (see Table 1). Each row comprises all data about all process steps of one instance of the process, e.g., details about machines settings. Moreover, the categorized value of the metric representing the optimization goal is added for completed process instances. For this purpose, the metric is selected in a preliminary step by an analyst who defines value ranges for undesired metric deviations. For example, lead times for the process of steel spring manufacturing which are higher than 27 minutes may be too high. This leads to a categorization of the metric with two values "OK" and "NotOK". It is important to remark that the metric refers to the entire process, not to a single process step.

For rBPO, a single instance of a process is analyzed during its execution according to the following two-step procedure, which is initiated every time a process step S_j with $j \in \{1, \ldots, n-1\}$ completes. Note that there is no need to run this procedure for the last step S_n in a sequential process.

(1) After the completion of step S_j, *real-time prediction* is run to forecast whether the entire process instance is likely to run into a metric deviation at the end, that is, whether the prediction reveals that the value for the metric will be "NotOK".

(2) If the latter is the case, there is a need for optimization to avoid the predicted metric deviation. Thus, *recommendation generation* is executed to deduce an action recommendation for the following process step S_m with $m = (j + 1) \leq n$.

In the following, we present an overview of real-time prediction in Sec. 3 and discuss recommendation generation as the central rBPO component in Sec. 4.

3 Real-Time Prediction

Real-time prediction comprises three steps, namely *definition of the data basis*, *mining model generation* and *mining model application* (see Fig. 1). In this section, we only highlight major technical aspects due to space limitations. A manufacturing-oriented discussion of real-time prediction issues can be found in our previous work [8]. With respect to the *definition of the data basis*, we refer to data about completed process instances in the PWH. From these process instances, we need (1) the attributes related to the already completed process steps of the running process instance for which we want to predict the metric value (*i400* in our example) and (2) the categorized metric value. In Table 1, this comprises the metric attribute and all attributes of steps one and two for process instances *i100* to *i300*.

For *mining model generation*, a suitable data mining technique has to be defined which uses the tailored data basis as training data. As we have to predict nominal values, classification techniques are employed [9]. Moreover, the generated mining model should be optionally presented to an expert user to enable him to comprehend the prediction and fine-tune parameters. Thus, the interpretability of the generated model should be comparatively high. In our previous work [8], we did a qualitative evaluation of major classification techniques with respect to their interpretability. To summarize, decision trees are comparably easy to understand and intuitively interpretable due to their graphical representation. Thus, we use decision tree induction as classification technique and focus on binary trees for the sake of enhanced understandability. The metric attribute represents the dependent attribute and the attributes of the set of completed steps $C_j = \{S_1, \ldots, S_j\}$ with $j \in \{1, \ldots, n-1\}$ are used as independent attributes for decision tree induction.

Finally, *mining model application* uses the decision tree to generate the prediction for the metric value. To this end, the data of the currently running process instance is used to traverse the decision tree and recommendation generation is started if the prediction reveals "NotOK".

4 Recommendation Generation

Recommendation generation deduces an action recommendation for the next process step in a running process instance. An action recommendation comprises several action items consisting of process attributes and a target value for each of them,

e.g., "Winding_Speed > 120". Thus, we base our concept on decision rules combining target values of process attributes. For this purpose, we generate decision trees which correlate the categorized metric value as a class label with selected attributes of selected process instances. Then, each path from the root node to a leaf node of the tree with the label "OK" represents a potential decision rule for a recommendation. We use decision trees to generate decision rules for the sake of comprehensibility for an expert user as described in Sec. 3. Another option could be to use association rule mining to deduce decision rules. Yet, this option lacks comprehensibility for the user. In addition, from a more technical point of view, since only association rules related to the metric attribute are relevant, computing *all* frequent item sets as commonly done in association rule discovery seems to be superfluous.

Recommendation generation encompasses the four sequential steps described in the following, namely *definition of the data basis, mining model generation, mining model analysis* and *recommendation processing* (see Fig. 1).

4.1 Definition of the Data Basis

The starting point for recommendation generation is the data basis provided by the PWH (see Table 1). In the following, restrictions on attributes and process instances used for recommendation generation for a process step S_m are discussed.

Regarding the *selection of attributes*, we generally assume that only attributes representing influencable factors like machine settings are considered, e. g., using a predefined filter. Thus, it is assured that recommendations only comprise directly applicable actions. Besides, attributes referring to completed process steps $C_j = \{S_1, \ldots, S_j\}$, are out of scope as they cannot be changed anymore. Moreover, attributes of all remaining process steps $R_m = \{S_m, \ldots, S_n\}$ with $m = j + 1 \leq n$ could be used to compare different recommendations for process step S_m with regard to their effects on later recommendations. Yet, this makes the evaluation of decision rules for the recommendation significantly more complex. Hence, we only use attributes of step S_m for recommendation generation for step S_m. In our example (see Table 1), these are amongst others "Machine_ID" and "Peening_Duration" for step S_3.

With respect to the *selection of process instances*, recommendations are derived using data about *completed* process instances because other running instances miss

Table 2. Data basis and restrictions for recommendation generation for process step S_3

Process Instance ID	Step1 Maschine ID	Step2 Material ID	Step2 Machine ID	Step3 Machine ID	Step3 Tool ID	Metric
i100	M12	MA43	M23	M33	T17	OK
i101	M12	MA43	M23	M33	T17	OK
i102	M12	MA43	M23	M33	T17	OK
i103	M12	MA44	M23	M34	**T18**	**NotOK**
i104	M12	MA44	M23	M34	T17	OK
i105	M12	MA44	M23	M34	**T18**	**NotOK**
i400	M12	MA44	M23			

Process restrictions:
- Material MA43 and Machine M33
- Material MA44 and Machine M34

the final metric value necessary for decision tree induction. Thereby, a decisive point is whether (1) all completed process instances are incorporated or whether (2) only completed instances which have the same attribute values as the currently running process instance are selected. To illustrate this point, Table 2 shows an exemplary data basis for recommendation generation for process step 3 in instance *i400*. Completed instances that have the same attribute value as the running instance are marked in blue (*i100-i105*). In general, there can be various implicit dependencies between process attributes due to process restrictions, which are not represented explicitly. For instance, certain materials used in step 2 may require specific machines in step 3.

In variant (1), one resulting recommendation based on the decision tree in Fig. 2 would be to use machine *M33* in step 3. Yet, this conflicts with the process restrictions because machine *M33* cannot be used with material *MA44*. In contrast, the decision tree in variant (2) reveals the one and only valid recommendation in this example, i.e., to use tool *T17*. This is because the dependency between material and machines can only be recognized by decision tree induction when solely instances *i103-i105* are used. Hence, we opt for variant (2), but we have to remark that a minimum amount of data about completed process instances with the same attribute values has to be available in order to recognize the dependencies.

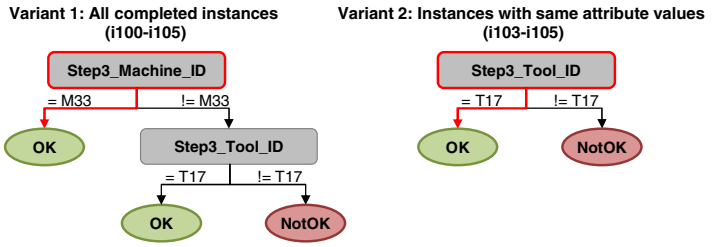

Fig. 2. Exemplary decision trees with different data bases

4.2 Mining Model Generation

Mining model generation comprises the generation of the decision tree on the data basis defined in Sec. 4.1. Below, we discuss the structure of the tree and its height.

With respect to the *structure of the decision tree*, we differentiate *binary trees* and *n-ary trees*, whereas the former has exactly two child nodes per node and the latter has arbitrarily many [9]. On the one hand, an *n-ary tree* reveals a *higher number of rules* due to the n-ary split, if we assume that a typical process attribute has more than two values. This increases the complexity of mining model analysis (see Sec. 4.3). On the other hand, the rules are supposed to be *more trustworthy* as they have a potentially *lower misclassification rate* compared to binary trees, when the maximum height of the trees is fixed. Yet, each decision rule in an n-ary tree is potentially backed up by *less underlying process instances* than a rule of a binary tree, if the maximum height is fixed. That is, the rules are derived from less process instances and thus are supposed to be *less significant*. Hence, for our approach, we opt for binary trees to

generate more significant recommendations and reduce the complexity of mining model analysis due to a smaller number of rules. Moreover, we decide to further restrict the trees by using only equal and not-equal relations for branches in order to speed up tree induction and simplify recommendations. That is, there are no subset restrictions on the branches. This makes recommendations more general and flexible. For instance, a recommendation may suggest employing all tools except tool 18.

With respect to the *height of the decision tree*, we define a maximum height depending on the concrete process and the number of process attributes in order to restrict the number of action items of a recommendation. Corresponding algorithms for decision tree induction and pruning to achieve the desired height are presented in Sec. 5.1.

4.3 Mining Model Analysis and Recommendation Processing

Based on the generated decision tree, mining model analysis comprises two steps, the *derivation of decision rules* and their *evaluation* in order to select the rule for the recommendation. For *rule derivation*, the tree is traversed from the root node to each leaf and each path which ends in a leaf node with the label "OK" results in a decision rule (see Fig. 3).

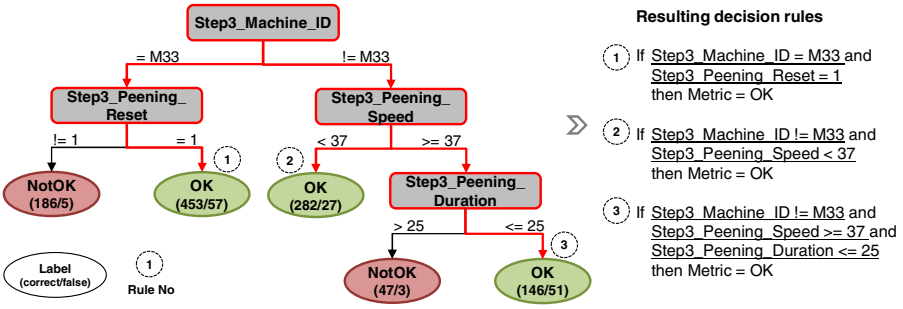

Fig. 3. Decision tree and resulting decision rules

Then, *rule evaluation* analyzes each rule according to the following four criteria:

The *misclassification rate q* is the percentage of process instances a rule does not classify correctly with respect to all instances covered by a rule, irrespective of their class label. The lower the misclassification rate, the higher is the trustworthiness of the rule. In general, rules are excluded that excess a threshold with, e.g., $q > 0.2$, depending on the concrete process.

The *percentage of underlying instances r* refers to the number of process instances a rule was derived from in relation to the total number of instances underlying the entire tree as training data. It represents the significance of the rule. Rules are excluded which do not apply to a minimum percentage of instances with, e.g., $r < 0.1$, depending on the concrete process.

The *length* of the rule *l* refers to the number of action items, that is, attribute-value-combinations of a rule. The shorter a rule, the easier it may be applied.

The *compliance with planned values c* refers to the correspondence of the action items of a rule with values defined during process planning, e.g., whether the recommended machine matches the planned one. Compliance *c* is defined as the percentage of matching attribute-value-combinations of a rule. The higher the compliance, the easier and faster a rule may be applied. Yet, it has to be remarked that there are attributes for which no production planning specifications are made and thus we do not define thresholds for *c*.

Table 3. Evaluation of decision rules

No	Rule	Missclassification (q)	Instances (r)	Length (l)	Compliance (c)	Score (t)
1	Step3_Machine_ID = M33 ∧ Step3_Peening_Reset = 1	11%	41%	2	50%	58
2	Step3_Machine_ID != M33 ∧ Step3_Peening_Speed < 37	9%	25%	2	0%	42
3	Step3_Machine_ID != M33 ∧ Step3_Peening_Speed >= 37 ∧ Step3_Peening_Duration <= 25	26%	16%	3	33%	-

For the final selection of a rule, we first filter all rules according to the defined thresholds for r and q in order to ensure a minimum quality of all rules. Then, we calculate a total score t with $0 \leq t \leq 100$ for each remaining rule and select the rule with the highest total score. To this end, a sub score is calculated for each criterion ranging from 0 to 25. These sub scores are summed up to the total score as follows:

$$t = ((1 - q) \times 25) + (r \times 25) + \left(\left(\frac{1}{l}\right) \times 25\right) + (c \times 25).$$

We equally weight all criteria but individual weights can be assigned depending on the concrete process. If there is more than one rule with the highest total score, they may be used alternatively or presented to the user. In the example (see Table 3), rule no 3 is excluded due to its excessive misclassification. Finally, total scores for rules no 1 and 2 are calculated and rule no 1 is selected for recommendation processing.

As the last step after mining model analysis, recommendation processing gets the selected decision rule as input and may either present it to the user in text form or feed it back in an operational system, e.g., a workflow management system, for further processing. For the sake of simplicity, we present the decision rule in human readable text to the user. In our example with rule no 1, a shop floor worker in manufacturing may receive the recommendation *"Use machine M33 and do a reset before peening"*.

5 Proof of Concept: Application in Manufacturing

Below, we provide a first technical proof of concept of rBPO based on a prototypical implementation for the manufacturing industry. Our *prototype* is based on the work of [10] and makes use of a relational implementation of the Manufacturing

Warehouse [6] as a PWH in IBM DB2. We use RapidMiner as a data mining tool and store decision trees in XML format. For decision tree induction, there is no generally valid algorithm as it heavily depends on the available data. For our prototype, we choose the C4.5 algorithm [11]. Alternatively, incremental decision tree algorithms [9] could be used to facilitate incremental model updating on the basis of new process instances. Real-time prediction and recommendation generation are implemented in Java.

For the *proof of concept*, we focus on the technical feasibility of rBPO and apply our prototype in an exemplary case, i.e., the manufacturing of steels springs for car motors as described in [7]. The process comprises sequential steps for winding, tempering, shot peening and testing of springs and employs different machines, e.g., winding automates and tempering furnaces. Based on a process study, we identified attributes of process steps and resources, e.g., winding speed and peening duration, as well as influence factors for metric deviations, e.g., machine settings. Then, we generated corresponding data on up to 100,000 process instances to populate the PWH.

On this basis, we *investigated recommendation generation* with respect to the requirements defined in Sec. 2. rBPO proved to proactively generate meaningful recommendations at process runtime (R2) focusing on metrics such as lead time and quality rate (R1). Thereby, it made use of operational and process manufacturing data (R3), e.g., from manufacturing execution systems and enterprise resource planning systems, integrated in the Manufacturing Warehouse. Decision trees were always generated on the entire data set to exploit the complete process history and realize adaptive recommendations (R4). The generated recommendations combined multiple action items (R5), e.g., on resources like machines, across different process steps.

In addition, we did *measurements* for multiple settings varying the number of process instances up to 100.000 instances on our test system (Windows Server 2008 R2, Core i7-2620M@2,7 GHz, 8 GB RAM). Each setting comprised 65 attributes across the process and was measured 5 times. For a setting with 100.000 process instances, our measurements reveal that data basis definition and mining model generation for real-time prediction take about 11 seconds on average. This is not critical as it can be done offline in advance. Mining model application takes less than one second which is suitable for online use at process runtime. For recommendation generation, data basis definition and mining model generation take less than 2 seconds on average. They have to be done online as the number of possible decision trees prevents an offline preparation. This is acceptable in typical manufacturing environments as there often is a delay between two manufacturing steps, e.g., due to transportation. Moreover, there is a significant potential for performance optimization, as the focus of our first proof of concept was on feasibility issues instead of pure response time. Finally, mining model analysis and textual presentation are done in less than 100 milliseconds.

To sum up, our first proof of concept demonstrates the technical feasibility and performance of rBPO. It proves that action recommendations can be proactively deduced at process runtime on the basis of a holistic process warehouse and that they can be generated quickly enough for an exemplary process environment. This provides the basis for an application in a real-world case in order to further evaluate the recommendations, e.g., comparing their effectiveness and comprehensibility.

6 Related Work and Evaluation

To structure related work, we differentiate three types of data analytics for process optimization [12]: *Descriptive analytics* focus on the manual and metrics-based analysis of completed processes as done in online analytical processing and reporting systems [13]. As opposed to that, *predictive analytics* forecast future process events. Recent approaches in process mining and business process intelligence [1, 14-17], e.g., for the prediction of metric values of running processes, belong to this category. Yet, all these approaches do not suggest concrete decisions but rather rely on the subjective judgment and analytical skills of the user to deduce improvement actions. In contrast, *prescriptive analytics* generate specific action recommendations to achieve a goal. That is, they build a bridge between pure analysis and actual optimization. In general, we observe two types of systems for prescriptive analytics: (1) recommender systems [18] using data mining techniques [9] and (2) expert systems [19] typically using rule-based, case-based and model-based reasoning techniques. We focus on data-mining-based concepts because expert systems require additional knowledge formalization and modeling and thus prevent a truly data-driven approach.

An initial approach towards prescriptive analytics for process optimization using decision trees is presented in [20]. It exploits a holistic process warehouse to generate decision trees predicting the performance of a new process instance. In case of a negative prediction, the instance is reconfigured before its execution. For the reconfiguration, the authors suggest to analyze the decision trees in order to deduce action recommendations. Yet, they do not provide any technical details on recommendation generation or decision tree evaluation and primarily focus on prediction issues.

Table 4. Comparative evaluation of rBPO

		rBPO	Pattern-based Optimization (PatOpt)	Recommendation-based Process Mining (RPM)	Risk-based Decision Support (RDS)
R1	Metrics-based Goal Definition	○	+	○	○
R2	Proactive Optimization during Process Execution	+	-	+	+
R3	Integration of Process Data and Operational Data	+	+	-	-
R4	Adaptive Recommendation Generation	+	-	+	+
R5	Multiple Action Recommendations	+	+	-	-

+ / ○ / **-** Approach fully/partially/does not meet(s) requirement.

To evaluate rBPO with respect to existing data-driven approaches, we did a qualitative comparison against the requirements defined in Sec. 2. For the comparison (see Table 4), we focus on the approaches of *pattern-based optimization (PatOpt)* [21], *recommendation generation using process mining techniques (RPM)* [22, 23] as well as the approach of *risk-based decision support (RDS)* [24] as these are the data-driven approaches most closely related to rBPO. PatOpt comprises a predefined catalogue of so called optimization patterns which encapsulate data mining techniques and

generate improvement recommendations, e.g., automating a certain decision activity to speed up the process. RPM focuses on operational decision support by recommending an action in order to optimize a metric, e.g., recommending the best resource for an activity. RDS predicts risks in terms of metrics deviations during process execution to provide decision support for certain actions, e.g., choosing the next process activity which minimizes process risks.

All four approaches support a *metrics-based goal definition (R1)*. Thereby, rBPO and RPM require the definition of one, possibly aggregated, target metric to be optimized, e.g., lead time. RDS focuses on risks in terms of metric deviations aggregated to a mathematical function. In contrast, PatOpt is based on the four target dimensions of process improvements, namely time, cost, quality and flexibility, and enables a multi-goal optimization. As opposed to the other approaches, PatOpt does not provide *proactive optimization (R2)* as optimization patterns are applied ex-post after process execution. With respect to the *data basis (R3)*, both rBPO and PatOpt are based on a holistic process warehouse integrating operational data and process data. RPM and RDS mainly focus on process data in an event log without explicitly integrating operational data on process subjects, e.g., machine data or master data on employees. Yet, we assume the integration of operational data to improve recommendation quality due to the augmented data basis. rBPO, RPM and RDS *adaptively generate recommendations (R4)* using data on past process executions. At this, rBPO employs classification techniques on warehouse data and RPM statistically evaluates the event log with traces of completed process instances. RDS employs decision tree induction on the event log to generate risk predictions. By contrast, the pattern catalogue of PatOpt constitutes a static collection of optimization best practices preventing adaptive recommendation generation. With respect to the generated recommendations, rBPO dynamically combines *multiple action items (R5)* of various types and PatOpt aggregates several optimization patterns. As opposed to that, RPM and RDS are less flexible and focus on a predefined type of action, e.g., to suggest the next activity to perform. At this, RDS provides only rudimentary decision support by predicting the potential risk for all possible actions without further concrete recommendations. All in all, rBPO goes beyond existing approaches by using a holistic data basis for adaptive recommendation generation in a fully data-driven manner.

7 Conclusion and Future Work

In this paper, we presented prescriptive analytics for recommendation-based business process optimization at process runtime. Our proof of concept underpins the technical feasibility and performance of our approach and emphasizes the importance of comprehensive data acquisition infrastructures, especially in manufacturing processes, to enable real-time process optimization. Moreover, it motivates the application in a real-world case to analyze recommendation quality and further refine decision rule evaluation. The realization of a closed loop feeding the recommendations back into a process control system is another aspect of future work.

References

1. Muehlen, M.Z., Shapiro, R.: Business Process Analytics. In: Vom Brocke, J., Rosemann, M. (eds.) Handbook on Business Process Management 2, pp. 137–158. Springer, Berlin (2010)
2. Kemper, H.-G., Baars, H., Lasi, H.: An Integrated Business Intelligence Framework. Closing the Gap Between IT Support for Management and for Production. In: Rausch, P., Sheta, A.F., Ayesh, A. (eds.) Business Intelligence and Performance Management, pp. 13–26. Springer, London (2013)
3. McCoy, D.W.: Business Activity Monitoring. Gartner Research Note (2002)
4. Melchert, F., Winter, R., Klesse, M.: Aligning Process Automation and Business Intelligence to Support Corporate Performance Management. In: Americas Conference on Information Systems (AMCIS), pp. 4053–4063. Assoc. f. Information Sys., New York (2004)
5. Radeschütz, S., Mitschang, B., Leymann, F.: Matching of Process Data and Operational Data for a Deep Business Analysis. In: Interoperability for Enterprise Software and Applications (IESA), pp. 171–182. Springer, Berlin (2008)
6. Gröger, C., Schlaudraff, J., Niedermann, F., Mitschang, B.: Warehousing Manufacturing Data. A Holistic Process Warehouse for Advanced Manufacturing Analytics. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2012. LNCS, vol. 7448, pp. 142–155. Springer, Heidelberg (2012)
7. Erlach, K.: Value stream design. The way to lean factory. Springer, Berlin (2011)
8. Gröger, C., Niedermann, F., Mitschang, B.: Data Mining-driven Manufacturing Process Optimization. In: World Congress on Engineering (WCE), pp. 1475–1481 (2012)
9. Han, J., Kamber, M., Pei, J.: Data Mining. Morgan Kaufmann, Waltham (2012)
10. Dapperheld, M.: Entwicklung analysebasierter Optimierungsmuster zur Verbesserung von Fertigungsprozessen. Master Thesis, University of Stuttgart (2013)
11. Quinlan, R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo (1993)
12. Evans, J.R., Lindner, C.H.: Business analytics. Decision Line 43, 4–6 (2012)
13. O'Brien, J.A., Marakas, G.M.: Management information systems. McGraw-Hill, New York (2011)
14. van der Aalst, W., Schonenberg, H., Song, M.: Time prediction based on process mining. Information Systems 36, 450–475 (2011)
15. Castellanos, M., Casati, F., Dayal, U., Shan, M.-C.: A Comprehensive and Automated Approach to Intelligent Business Processes Execution Analysis. Distributed and Parallel Databases 16, 239–273 (2004)
16. Grigori, D., Casati, F., Castellanos, M., Dayal, U., Sayal, M.S.M.: Business Process Intelligence. Computers in Industry 53, 321–343 (2004)
17. Kang, B., Lee, S.K., Min, Y.-B., Kang, S.-H., Cho, N.W.: Real-time Process Quality Control for Business Activity Monitoring. In: Computational Science and Its Applications (ICCSA), pp. 237–242. IEEE, Los Alamitos (2009)
18. Jannach, D., Zanker, M., Felfernig, A., Friedrich, G.: Recommender systems. Cambridge University Press, New York (2011)
19. Giarratano, J.C., Riley, G.: Expert systems. Thomson Course Technology, Boston (2005)
20. Grob, H.L., Bensberg, F., Coners, A.: Rule-based Control of Business Processes - A Process Mining Approach. Wirtschaftsinformatik 50, 268–281 (2008)
21. Niedermann, F., Radeschütz, S., Mitschang, B.: Business Process Optimization Using Formalized Optimization Patterns. In: Abramowicz, W. (ed.) BIS 2011. LNBIP, vol. 87, pp. 123–135. Springer, Heidelberg (2011)

22. Schonenberg, H., Weber, B., van Dongen, B.F., van der Aalst, W.M.P.: Supporting Flexible Processes through Recommendations Based on History. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 51–66. Springer, Heidelberg (2008)
23. van der Aalst, W.M.P., Pesic, M., Song, M.: Beyond Process Mining: From the Past to Present and Future. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 38–52. Springer, Heidelberg (2010)
24. Conforti, R., de Leoni, M., La Rosa, M., van der Aalst, W.M.P.: Supporting Risk-Informed Decisions during Business Process Execution. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 116–132. Springer, Heidelberg (2013)