Why Are Process Variants Important in Process Monitoring? The Case of Zalando SE

Matthias Schrepfer, Matthias Kunze, Gunnar Obst, and Juliane Siegeris

Abstract

(a) **Situation faced**: Business process models serve various purposes. As precise documentations of an implemented business processes, they provide inputs with which to configure process monitoring systems, enabling the specification of monitoring points and metrics. However, complex business processes have a quantity of variants that can impede the activation of process monitoring. To mitigate this issue, we seek to reduce the number of process variants by performing behavioral analyses.

(b) **Action taken**: Variants of a business process originate from points in the process model where the control flow might diverge, such as at decision gateways and racing events. We systematically identify the underlying semantics to choose from a set of alternative paths and characterize the resulting variants. This effort offers the opportunity to reduce the variability in business processes that is due to modeling errors, inconsistent labeling, and duplicate or redundant configurations of these points.

(c) **Results achieved**: For a sub-process of an order-to-cash process from the e-commerce industry, we discovered 59,244 variants, of which only 360 variants lead to a successful continuation of the process. The remaining variants cover exception handling and customer interaction. While these variants do not lead to a successful outcome and might not qualify for the "happy path" of this process, they are crucial in terms of customer satisfaction and must be monitored and controlled. Using a set of methods (actions taken),

M. Schrepfer (✉) • M. Kunze • G. Obst
Zalando SE, Mollstr. 1, 10178 Berlin, Germany
e-mail: matthias.schrepfer@zalando.de; matthias.kunze@zalando.de; gunnar.obst@zalando.de

J. Siegeris
HTW Berlin University of Applied Sciences, Berlin, Germany
e-mail: juliane.siegeris@htw-berlin.de

J. vom Brocke, J. Mendling (eds.), *Business Process Management Cases*,
Management for Professionals, DOI 10.1007/978-3-319-58307-5_23

we reduced the number of variants to 11,000. These actions reduced overhead in the process and normalized decision labels, thereby significantly increasing the process model's quality.

(d) **Lessons learned**: We elaborate on the impact of variants on the configuration of a process monitoring system, and show how the number of model variants can be significantly reduced. Our analysis shows that the semantic quality of the process model increases as a result. This reduction effort involves a structured approach that considers all variants of a business process, rather than focusing only on the most frequent or most important cases.

1 Introduction

Business process management is an established discipline that is widely used in industry. Many companies focus on established methods to design, analyze, control, and optimize their business processes and to ensure high levels of customer satisfaction and close alignment with IT systems (cf. Hammer 2010). Rapidly growing multinational companies in the e-commerce sector in particular must overcome challenges in business process management in order to scale up their businesses and reach ambitious business goals, so business processes in this sector are largely automated. Setting up consistent and scalable process monitoring and process controlling helps firms to detect problems, derive remediating actions and to address these problems quickly.

1.1 Business Process Management at Zalando

Business process management found its entrance into Zalando in 2012, when the company set out to document its core processes in a structured way. Because of the company's rapid growth, we decided to develop and tailor to our needs our own ERP system, Zalando E-Commerce Operating System (ZEOS). For the requirements specification of this system and to ensure proper alignment between the business and IT, all departments involved contributed to the precise documentation of the relevant business processes using BPMN. Over time, increasing numbers of processes in Zalando's value chain were documented and integrated into the company's process landscape.

One year later, we began to use the documented business processes for operational tasks. We experimented with a self-developed process engine to automate our core order processes, which led eventually to the integration of an open source BPM engine and the first fully automated business process's going live early in 2014. Since then, we have continuously increased the automation of our processes.

We also found significant value in detecting anomalies in the execution of our processes, including non-automated and hard-coded behavior. We devised an approach that enables business processes to be monitored using real-time event

data that are provided from all IT systems involved. Using a highly scalable architecture, we can monitor hundreds of thousands of orders per day and provide early warnings and near real-time detection of anomalies for our end-to-end core processes. The data created remains available for ex-post analysis and as a basis for continuous improvement.

BPM has become one of the driving forces and key factors of success in Zalando's endeavor to become a widely used platform that connects people with fashion beyond its core business.

1.2 The Role of Process Monitoring

Enabling process monitoring requires that process models contain all of the business logic required by underlying business scenarios and that they consider processes across the IT landscape and organizational boundaries. Doing so typically results in a large number of detailed and complex process models that capture all possible cases. While the creation of models of high syntactic and semantic quality is challenging in practice, it is required for process monitoring and it bridges the gap between business and IT, so it builds the basis for process execution, compliance checking, and continuous improvement.

Effective process monitoring ensures that business goals are met by continuously checking the state of and performance of business processes (Dumas et al. 2013). This monitoring includes detecting process problems and raising warnings and alarms when there are problems or deviations. While this monitoring may sound straightforward given detailed process models, it is subject to several constraints in practice. What makes process monitoring so complicated?

To detect and resolve problems with a business process rapidly, all process instances must be monitored. In the e-commerce setting of a large organization, where core processes are highly automated and executed many times, the number of process instances quickly rises beyond 100,000 in 24 h. It is critical that the productive and efficient operation of every one of this instances persists in a highly competitive environment, which is already a technical challenge in terms of the scalability of the monitoring system.

Furthermore, the more complex the process, the more complex the process monitoring because all process variants must be treated separately. Here, the term *process variant* refers to all possible paths in a process model that must be monitored (Dumas et al. 2013). Different process paths are triggered by parameters like the shipping or payment method chosen. Each parameter yields an individual process flow in such a way that individual values, such as those for payment methods like credit card and invoice, are handled properly. Business and IT users must know whether all of these flows are executed properly in order to ensure conformance with the process. However, each variant that is monitored should be treated separately, which results in the need for an enormous effort to set up the monitoring system.

The lower the number of process variants in a process, the easier its activation. In this chapter, we present approaches to analyzing the parameters that trigger process variants in an effort to reduce the number of process variants. By analyzing process

variants, we also show opportunities to increase the quality of process models from a semantic point of view. Our goal is to reduce the effort in and increase the efficiency of activating process monitoring.

2 Situation Faced

We illustrate our approach using part of an order-to-cash process, a real-world example depicted in Fig. 1. The part of the process we investigate starts with the placement of a customer order and ends with the decision concerning to which warehouse the shipment of the ordered goods is assigned. The business process,

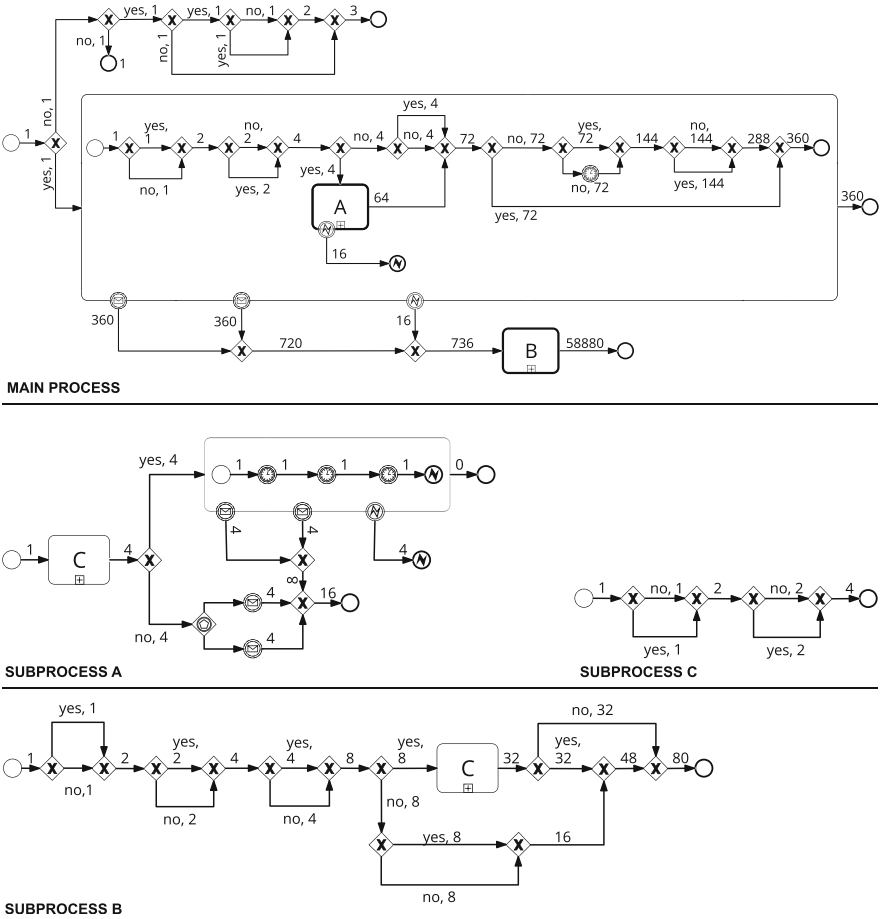

Fig. 1 Order-to-cash main process and subprocesses: The process model shows only the branching structure for our order-to-cash process, as we removed activities and labels. We annotated control flow edges with the number of variants that can pass through these points in the process model

modeled using BPMN, consists of one parent process and three subprocesses. The original models consist of 20–100 elements and contain both basic and advanced process modeling structures, such as error-handling, process hierarchy, and attached boundary events. In our case, all process steps are executed sequentially; that is, the business process contains no concurrency.

In Sect. 3.2, we discuss in greater detail how the number of process variants—59,244—were computed. Subprocess B (Fig. 1) can be initiated by 736 variants, and the subprocess itself creates 80 variants as continuations of each of the incoming variants. Hence, the number of variants multiplies when subprocesses are included, leading to 58,880 variants after the subprocess is completed.

Along the process, we established several measurement points for which our monitoring system records the time and process data. Our monitoring solution allows us to compute the time period between two measuring points continuously to compare these metrics with threshold values, and to visualize the current and historic performance of a business processes. If the threshold value for a given metric is exceeded for a certain number of instances, the system notifies affected personnel.

As we show in the remainder of this chapter, the number of variants is an upper bound that can be significantly reduced.

3 Action Taken

The methods presented in this chapter refer to the discovery of process variants in business process models. The literature advocates two approaches: The multi-model approach uses a number of related process models to capture variants, typically as a result of manipulating one central reference model (Li et al. 2011; Sakr et al. 2011), while the single-model approach consolidates all possible variants into one process model that offers different configurations for a particular variant (Hallerbach et al. 2010; Rosemann and van der Aalst 2007). In the second approach, some gateways are marked as configuration points, where different variants follow different branches. Still, in both cases, a process variant is a complete business process model that includes control-flow branching structures.

In this chapter, variants are understood as distinct sequences of activities and events, similar to the notion of traces in process mining (cf. Dumas et al. 2013; van der Aalst 2011). Process mining analyzes the logs of business process executions and strives to define process models by reverse-engineering ordering relationships between activities and detecting where a path in a process might diverge. In contrast to process mining, our approach does not use process logs as the basis on which to generate a process model but starts with the model itself to reveal all possible variants. The number of variants is related to the cyclomatic number of programs (McCabe 1976; Myers 1977). However, in our case, iterations of the same process model fragment are also considered individual variants.

Based on the variants discovered, this work seeks to improve the quality of a process model by reducing the number of variants and increasing the consistency within it. Model quality has been the focus of a wide range of research, as an overview of the factors that affect process models' quality shows (Mendling et al. 2009). Our primary focus is on the process models' semantic and pragmatic quality (Reijers et al. 2010).

One particular issue that has not been addressed in the literature is the consistency of the configuration of points in business process models, where process execution diverges. We refer to these points as *trigger parameters for variants of a process model*. For instance, if two distinct exclusive-choice gateways model the same decision, they should be labeled identically. The following sections present our approach to discovering, characterizing, and reducing process variants and to normalizing choices within a process model.

Other proposals related to increasing the quality of process models include Mendling's Seven Process Modeling Guidelines (Mendling et al. 2010), which introduces rules based on empirical research to keep process models simple, consistent, and easily comprehensible. While these guidelines can improve a single model and reduce its cognitive complexity, refactoring of process models (Weber and Reichert 2008) strives to increase the consistency among several models in a collection, such as through consistent labeling of activities across all models. Weber and Reichert (2008) use of the term *variants* uses a different meaning than we use here.

Many of these approaches to increasing process models' quality had already been applied when our business processes were modeled, such as the labeling of objects and the extraction and linking of common subprocesses. Figure 1 illustrates the linking of processes, subprocess C is linked to processes A and B. However, these approaches do not change the semantics of a business process model but their organization, so they have no impact on a process model's number of variants.

3.1 Variants in Business Processes

In contrast to the related work discussed in the previous section, where process variants refer to different versions of a complete business process model, we define a process variant as a class of process instances:

> A **process variant** is a complete and unique sequence of activities, events, and decisions carried out in compliance with a business process model. Every process instance of this model belongs to exactly one process variant.

Zalando's order-to-cash process is executed among a number of independent and distributed software systems, each of which adds fragments and execution alternatives to the process. Our definition of a business process variant embraces this aspect of distributed IT environments and captures one variant of the overall

process as a particular ordering scenario. Variants must be complete with regard to the start and end of the process.

3.2 Identification of Process Variants

Having defined the term *process variant*, we ask how variants can be identified. Business process mining offers a straightforward solution to the identification of unique variants—examining a process log—but in our scenario, such a log is not available, as our goal is to set up a monitoring solution prior to the rollout of a business process. Even so, it is possible to derive process variants from a process model if it is normative and sufficiently detailed, that is, if it is on an executable level. Essentially, all model constructs that yield alternative outcomes lead to a set of process variants such that each alternative adds another process variant. In the case of BPMN, such constructs can include exclusive gateways and interrupting boundary events.

Our approach to computing the number of variants in a process model is based on Sadiq and Orlowska (2000), who present an approach to identifying behavioral anomalies in sequential process models by iteratively eliminating paths in the model that are correct. For instance, a set of n alternative paths that are split and joined in a well-structured fashion are reduced to a single path. If the remaining model only contains single paths, then the original model was correct. Models that show deadlocks or lack of synchronization cannot be reduced completely.

In our case, the process models underwent a verification process a priori, so they are considered to be correct, but we reused the iterative reduction technique to identify variants in process models. For every reduction, we counted the number of variants that were created by the reduced fragment. Given the aforementioned set of alternatives, we would infer n variants and then annotate the number of variants to the outgoing control flow sequence. The number of variants is the input for the next fragment to be reduced. Subprocess C in Fig. 1, shows two fragments, each with two alternatives, which results in an overall count of four variants. Similarly, hierarchical decomposition in process models—that is, the use of subprocesses—adds significantly to the number of the business process's variants.

This method for deriving process variants is applicable only with well-structured, sequential process models (van der Aalst et al. 2002). Furthermore, the interwoven execution of parallel paths quickly explodes the number of process variants (Valmari 1998). In our case, the prerequisites of having well-structured models and sequential execution of activities apply because all parts of the end-to-end business process are carried out sequentially by different IT systems.

3.3 Characterizing Process Variants

A small number of process variants—perhaps around 500—is not problematic for process monitoring, as not every activity, event, or decision is tracked by a

monitoring system. In our example, we computed the number of variants according to the method described above for the first part of our end-to-end business process, which resulted in 59,244 process variants, a number that becomes unmanageable if our monitoring system is configured manually. The high number of variants was not expected, but it confirmed our initial concerns about process complexity.

As shown in Fig. 1, only 360 variants are successful—that is, only 360 lead to the order-to-cash process's continuing to the next subprocess—which is often referred to as "the happy path." Comparing this number with the overall number of variants shows that most variants address deviations from the happy path, and a semantic analysis shows that almost all other variants cover parts of the process for error-handling and customer interaction, such as order cancellations triggered by customers.

3.4 Reducing Process Variants

With 59,244 process variants for only part of an end-to-end business process, we sought to determine why variants are triggered. To this end, one goal was to remove variants whenever possible to ease process monitoring. Such a large number of variants may also be a sign of the potential to increase the quality of our process model. In this section, we report on the approaches to reducing variants that we identified by studying the process model and related information.

3.4.1 Zero Variants

One of the first reasons that process variants are triggered is paths in the process model that can never occur, which we call *zero variants*. Although we reviewed all of our process models prior to the variant analysis, our review was flawed from a semantic point of view. An example from subprocess B (Fig. 1, and shown in detail in Fig. 2) internally handles an error before escalating that error to the parent scope.

The process path identified by the outgoing blank end event of the subprocess is unreachable because the subprocess always terminates with an error event. An analysis of this path indicates that it increases the effort required in understanding the model and may lead to misinterpretations. Hence, all paths with zero variants must be refactored to increase model quality.

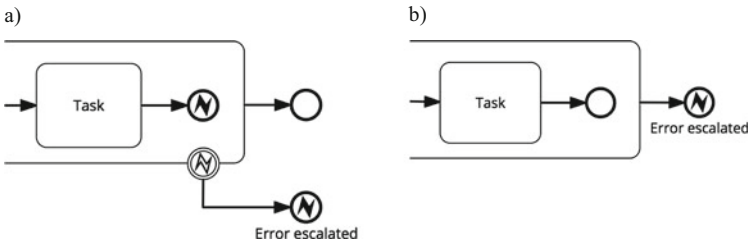

Fig. 2 Zero variants. (**a**) Original model (**b**) Refactored model

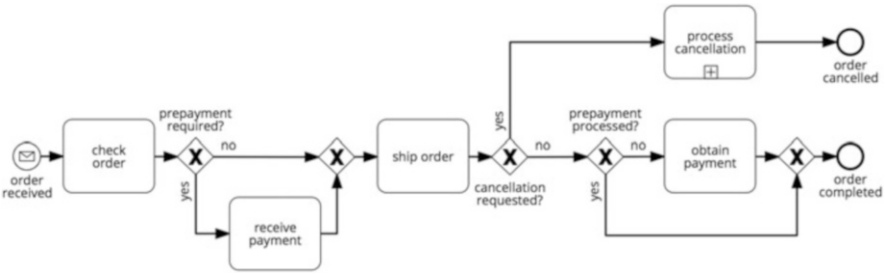

Fig. 3 Non-normalized decision

3.4.2 Duplicate Variants

The semantic analysis of process models—that is, the matching of model elements like activities, events, and decisions to their counterpart in our business—revealed a second opportunity to improve the quality of the process model and reduce the number of variants. Choices from among a set of alternatives in the process models are frequently not made completely independently; that is, the choice made at one point may depend on a choice made earlier in the course of executing the process. Figure 3 illustrates this factor with a fictitious example.

The business process shown in Fig. 3 contains a number of decisions. Two of them, *prepayment required* and *prepayment processed*, refer to the point at which the payment for an order is carried out. If the customer chose a form of prepayment, payment will be carried out before the order is shipped, but if the customer did not choose prepayment, the payment must be obtained after the order is shipped.

Looking only at the model, the process produces six variants, one for each combination of alternative paths. Taking into account the actual implementation of these decisions, we discovered that both decisions regarding payment are based on the customer's chosen method of payment. From a set of payment methods, one part qualifies for prepayment, whereas the remaining part does not. Hence, these two decisions are based on the same semantic context, so there are actually only four variants in the process model.

We introduce *trigger parameters*, *configuration parameters*, and methods to identify such dependencies and resolve them.

A **configuration parameter** (CP) is a variation dimension—that is, a set of values that denote alternatives.

A **trigger parameter** (TP) denotes a variation point in the process model that uses CPs that specify how to choose from among alternatives.

TPs characterize variants based on either conditions, such as at an XOR gateway, or based on events, such as at attached intermediate boundary message events, so they correlate process variants with elements of the process model. To identify duplicates, all TPs are listed separately with unique IDs, the condition of a gateway or the name of the event, and the process in which it is contained. Then a number of

checks are carried out to identify duplicate and redundant TPs, duplicate trigger configurations, and merging of events.

3.4.3 Duplicate Trigger Parameters

Duplicate labels of TPs are identified and marked, but corresponding points in the model are not yet refactored, as there is a chance of finding replicas of the TP, and duplicate labels do not necessarily imply duplicates, as the CPs for these TPs must also coincide.

Currently, the duplicate detection uses only a simple string comparison, and language processing is done by a domain expert to identify duplicates. In the future, natural language processing could assist (cf. Leopold 2013). A second quality check focuses on labels assigned to TPs—that is, their corresponding conditions and event names. TPs' labels should comply with a style that ensures that readers can quickly comprehend the semantic information. As a labeling style, we focus on a best-practice approach. See, for instance, Mendling et al. (2010):

> **for events**: object + past perfect verb
>
> **for gateways**: a question attached to the gateway; condition expressions must be an answer to the question stated at the corresponding gateway; both question and answers are brief and precise.

The result of these checks is stored along with the specification of TPs. Labels that violate these standards are marked for refactoring. However, refactoring labels is still postponed because of the need for additional checks. Moreover, not all labels may be refactored, as some are used in a close business-IT alignment. Thus, some labels, particularly event names, are also used in the implementation of IT systems and are used for monitoring. Hence, best practices may be neglected to keep models and implementations in sync.

3.4.4 Redundant Trigger Parameters

The next check focusses on TPs that can be eliminated, which will decrease the number of variants. The check determines whether a process model can be refactored in such a way that the TP is eliminated without changing the process semantics, as otherwise the process logic would change. This task is performed by process experts and domain experts to ensure consistency. A reduction in the number of TPs improves the model's comprehensibility and increases its quality.

Figure 4, which shows an excerpt of our example process, illustrates the check for redundant TPs, with one fragment showing a split XOR-gateway that corresponds to a TP. Assuming that only a single variant is provided as input, there will be two variants—one that includes the timer event and one that does not. The question addressed at the split gateway and the condition at the intermediate timer event are similar, so from a semantic point of view, the condition is checked twice: If the time has not progressed far enough, the process will wait for it using a timer event. An equivalent logic is also shown in Fig. 4, where only the

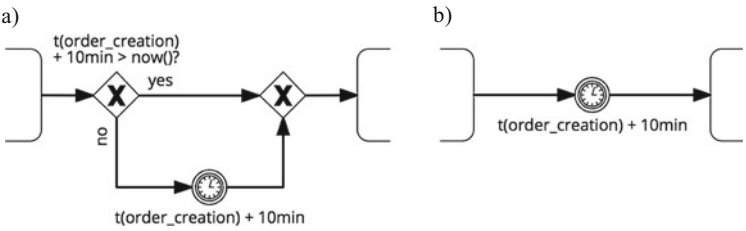

Fig. 4 Redundant trigger parameters. (**a**) Original model (**b**) Refactored model

intermediate timer event is used. The TP is avoided, reducing the complexity of the model and eliminating another variant.

3.4.5 Duplicate Trigger Configurations

Identification and documentation of duplicate and redundant TPs are the first steps toward understanding why variants occur. Information on TPs already supports the analysis of variants, and actions can be derived from the analysis results that can lead to removal of TPs and, consequently, decrease the number of process variants and increase the model's quality.

The next step toward reducing process variants is analysis of CPs, which are used to split up a business context, such as payment methods in the order-handling process. CPs are bound to TPs by assigning information about the business context. A TP that is linked to a specific process model element determines the process' behavior based on the information from a CP.

In the example shown in Fig. 3, the decision concerning which path is chosen is based on the customer's choice of a payment method for two out of three gateways. However, the information in the process model alone is not sufficient to determine whether these decisions are made under identical conditions. In practice "yes" and "no" do not determine which path to choose; instead, the actual selection of the payment method is required. In the future, it could payment methods that require both a prepayment before and a final payment after the shipment could be possible, but here CPs come into play, as they bind the actual conditions of TPs to values from a business context. For each CP, we store a unique ID, its name, and all of its unique values.

We distinguish among three types of CPs, which indicate how the value of the parameters is determined.

design-time CP: The parameter is static; for example it is used to configure an IT system.

a priori run-time CP: The parameter is determined at the creation of a process instance and does not change; for example, it is based on the received order that triggered a process instance.

live run-time CP: The parameter is determined during the run of the process instance; for example, it is the outcome of a human task or a value that is computed by an IT system.

These types of CPs are closely related to process variants. (See Sect. 3, where one variant is comprised of a complex model, including decisions.) Here, the *design-time CP* and *a priori run-time CP* can be used to exclude certain variants (in our definition) from the process model when a process is instantiated. *Design-time CPs* are comparable to configuration points in complex variant definitions.

In our example, the CP "payment method" consists of the values credit card, Paypal, invoice, and cash-on-delivery; the first two options are prepayment options, and the last two are payments after shipping. The type of CP is *a priori run-time* because it is based on the customer's choice of a payment method recorded in the incoming order. The CP is used for the TP of two gateways in the process model of Fig. 3.

The binding of a CP's values to the conditions of the outgoing paths of the gateway bridges the gap between domain knowledge, that is, the actual process execution and TPs in process models. Using this connection, we can identify TPs that use the same CPs. In combination with the search for duplicate TPs, we can normalize the process model by making decisions that are identical or similar consistent in their labelling.

First, we verify that all duplicate TPs are actually duplicates, that is, that they use the same CP values for the decision. If this is not the case, the labels in our model are ambiguous, which should be resolved by renaming one or both of the TPs in the model. Duplicate TPs with identical CP values are recorded as actual duplicates on the list of variants and are later refactored manually.

Second, we analyze TPs that have the same CP values. If a number of choices with semantically identical TPs and CPs occur in one process instance—that is, if they lie on a common path in a process model without concurrency—then a decision for one choice determines the decision in other choices, which reduces the number of variants.

3.4.6 Merging of Events

During our analysis of the process model, we found another opportunity to reduce the number of process variants. In some cases, variants were triggered by two or more message-receive events that indicated the same business trigger but differed in the data payload. Such variants can be treated as a single instance as long as some constraints are met:

> All events must have the **same** scope; for example, boundary events are attached to the same scope.

> The control flow of all events must be merged **directly** succeeding the message events.

These constraints ensure that different events, such as different messages, do not have different effects on the state of the business process. The decision concerning whether events can be merged must be made by domain experts, who must agree in terms of whether some events cannot be distinguished later on in the monitoring system. If they do not agree, the monitoring system must be configured in such a way that all events are monitored.

4 Results Achieved

We have presented several approaches to decreasing the number of process variants. We computed the number of variants for the first part of our end-to-end business process as 59,244, an unmanageable number if the monitoring system is configured manually.

As Fig. 1 shows, only 360 variants are successful—that is, only 360 lead to the continuation of the order-to-cash process. Comparing this number with the overall number of variants demonstrates that most variants address deviations from this "happy path." A semantic analysis shows that almost all other variants cover parts of the process for error-handling and customer interactions, such as order cancellations triggered by customers.

With the help of these approaches, we reduced the number of variants to 11,000. Because of the process hierarchy, the number of variants on the happy path dropped from 360 down to 120, significantly easing monitoring. This immense reduction was triggered by optimizing only two local areas, and the changes applied to the process model also reduced overhead, normalized decision labels, and increased the quality of our process model significantly.

4.1 Handling Zero Variants

Refactoring took place in the handling of "zero variants" and was performed without changing the process semantics from a business point of view. For instance, the quality of the process model in Fig. 2b increased without changing the number of variants. However, such may not be the case for process models that stem from other scenarios. We tested the approach presented here, and for several models the number of variants did change, even increasing in some situations, such as when the model contained boundary events. Hence, the number of variants must be computed again after model refactoring.

4.2 Handling Duplicate Trigger Parameters

Twenty-three duplicate TPs were detected in the first part of the order-to-cash process. An evaluation of these TPs for best-practice standards revealed that only four complied with best-practice naming standards, a very low success rate (~17%). Another 5 of the remaining 19 TPs could not be renamed because of their reuse in IT systems (~22%). The last 14 TPs (~61%) were adjusted according to best-practice naming standards.

4.3 Redundant Trigger Parameters

Although process models are checked for quality, TPs may be modeled in a redundant way, so our approach detects these triggers and applies remediation. The

number of variants that are due to TPs can multiply throughout the process, such as when there are subprocesses, so saving even one variant locally can reduce the global number of variants significantly.

In fact, using our approach to eliminating redundant TPs decreased the number of variants to approximately 36,000, a 39.3% reduction. We had not been aware that such large reductions could be achieved in practice, but the effect is so large because the removal of a single TP may affect the complete process hierarchy. When processes or parts of processes are scoped by boundary events, the decrease in local variants might also significantly decrease variants on a global scale, as our example shows.

4.4 Duplicate Trigger Configurations

Variants are created based on TPs, so the evaluation of process variants also includes determining the configuration of those TPs, revealing the underlying conditions. Upon applying our approach, we found two TPs tagged with "gift voucher" and "gift voucher bought," suggesting a potential duplicate. However, the values of the corresponding CPs revealed that the first one addressed the payment of an order using a gift voucher, whereas the second one incorporated the purchase of a gift voucher. This example highlights the importance of verifying duplicates using CPs.

In order to refactor the process model, one must determine why the same business context, that is, the set of CPs, is applied to TPs with different labels. In our case, the main reasons were errors in process models and a gap or mismatch between modeling and interpreting business information. Process experts and domain experts clarified how to remediate this discrepancy by deciding upon the TP and updating the label of the other to match the context if necessary. Then duplicate TP entries can safely be eliminated, which reduces the number of variants.

Another example is that of TPs tagged with "articles exist" and "article exists." The first conveys the impression that *all* articles must be available, whereas the second suggests only *one* article was sufficient. However, consulting domain experts and CPs led to the decision that the two TPs are identical, and one was renamed accordingly.

Out of the 23 TPs identified initially, 4 were removed because of duplicate TPs or duplicate CPs, which increased the consistency and reduced the ambiguity of the process model. The labels now better fit the domain knowledge. Even more important is that readers of the process model can determine which domain information is used in TPs and how variants are triggered. The semantic binding helped to increase the process model's quality significantly, even though the number of variants was not decreased in the real-world example for this approach.

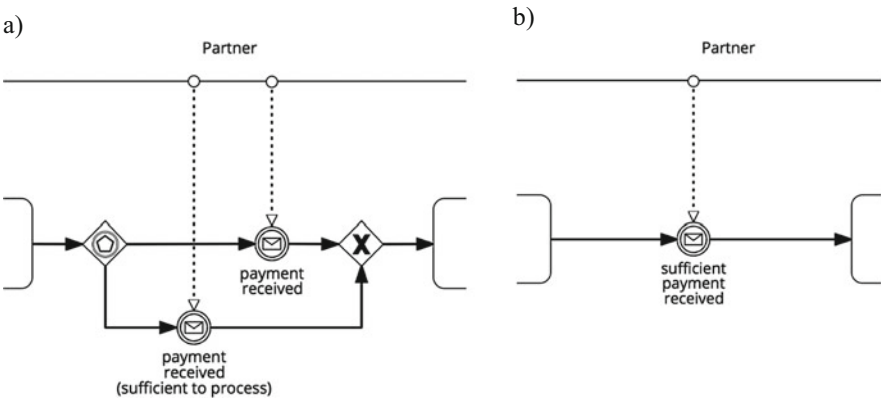

Fig. 5 Merging of events. (**a**) Original model (**b**) Refactored model

4.5 Merging Events

With regard to process monitoring, it is reasonable to merge events that fulfill the requirements stated in Sect. 3. In Fig. 5, we identified two variants that were triggered by two message events, respectively, of which only one is processed according to the event-based gateway that precedes these events. Both events indicate an incoming payment, but subtle differences led to their distinction in the model. After the issue was disclosed following the conditions above, domain experts confirmed that it was possible to merge the events for the purpose of monitoring. The original model was not changed in this case, and the refactored model is used only to configure the monitoring solution. This approach has the disadvantage of requiring that two models are in synchronization. Still, in many cases, the benefit of reducing variants outweighs the cost of maintaining two models.

5 Lessons Learned

We introduced an approach to characterizing and reducing variants in business process models based on the notion of TPs and CPs that provide insight into the data and logic that is applied when control flow diverges within the process model. Here we summarize our experiences and the insights gained during our study.

No Exclusion of Variants Implementing monitoring solutions often requires focusing first on important parts of a business process. Although we may not monitor all variants of a process, we cannot exclude any part of the process model from monitoring a priori. Even domain experts typically do not know

which variants are infrequent without a proper throughput analysis, so it is virtually impossible to identify the most important parts upfront. The only chance is to enable monitoring in such a way that monitoring considers all variants. If variants are excluded from monitoring, experience has shown that process problems are detected late, if at all. One should carefully judge whether and why to exclude variants from monitoring.

Bias for the Happy Path In process analyses it is common to concentrate on the happy path first, but it is not sufficient to just focus on the most common and expected behavior if complete monitoring of a process is the goal; 100% of the process variants must be monitored. One important observation we made during the identification of process variants is that happy paths typically contain only a minor portion of all variants, so it is likely that most of the process errors or problems are ignored in the happy path, although they must be considered as well.

Automation for Analyses Currently, all of our approaches are based on manual work. Because of the manual effort and likelihood of errors, we sought to reduce the manual work in favor of automated solutions. For instance, we researched the automatic discovery of variants and focused on recommendations for reducing variants. We also considered concurrent activities on different process paths. Consequently, different orderings of interwoven activities must not be considered as distinct variants if they follow along the same paths in the process model. This conclusion is contrary to the notion of process variants that originate from process-mining scenarios, which has also been applied here.

High Number of Process Variants We did not expect such a high number of variants from applying our approaches to the first process, but they confirmed our initial concerns of process complexity. We used several process models to verify our approaches, and in all situations the final number of process variants was surprisingly high. Even domain experts and model experts were surprised, but they ultimately understood why reducing the number of variants is needed in order to activate process-monitoring solutions quickly and efficiently.

References

Dumas, M., La Rosa, M., Mendling, J., & Reijers, H. (2013). *Fundamentals of business process management*. Berlin: Springer.

Hallerbach, A., Bauer, T., & Reichert, M. (2010). Capturing variability in business process models: The provop approach. *Journal of Software Maintenance, 22*(6–7), 519–546.

Hammer, M. (2010). What is business process management? In *Handbook on business process management: Introduction, methods and information systems* (Vol. 1, pp. 3–16). Berlin: Springer.

Leopold, H. (2013). *Natural language in business process models – Theoretical foundations, techniques, and applications*. Lecture notes in business information processing (Vol. 168). Berlin: Springer.

Li, C., Reichert, M., & Wombacher, A. (2011). Mining business process variants: Challenges, scenarios, algorithms. *Data Knowledge Engineering, 70*(5), 409–434.

McCabe, T. (1976). A complexity measure. *IEEE Transactions on Software Engineering, 2*(4), 308–320.

Mendling, J., Recker, J., & Reijers, A. (2009). *Process modeling quality: A framework and research agenda* (BPM Center Report, BPM-09-02).

Mendling, J., Reijers, H., & van der Aalst, W. (2010). Seven process modeling guidelines (7PMG). *Information and Software Technology, 52*(2), 127–136.

Myers, G. (1977). An extension to the cyclomatic measure of program complexity. *SIGPLAN Notices, 12*(10), 61–64.

Reijers, H., Mendling, J., & Recker, J. (2010). Business process quality management. In *Handbook on business process management* (1st ed., Vol. 1, pp. 167–185). Berlin: Springer.

Rosemann, M., & van der Aalst, W. (2007). A configurable reference modelling language. *Information Systems, 32*(1), 1–23.

Sadiq, W., & Orlowska, M. (2000). Analyzing process models using graph reduction techniques. *Information Systems, 25*(2), 117–134.

Sakr, S., Pascalau, E., Awad, A., & Weske, M. (2011). Partial process models to manage business process variants. *International Journal of Business Process Integration and Management (IJBPIM), 6*(2), 20.

Valmari, A. (1998). The state explosion problem. In *Lectures on petri nets I: Basic models, advances in petri nets*. Lecture notes in computer science (Vol. 1491, pp. 429–528). Berlin: Springer.

van der Aalst, W. (2011). *Process mining – Discovery, conformance and enhancement of business processes*. Heidelberg: Springer.

van der Aalst, W., Hirnschall, A., & Verbeek, H. (2002). An alternative way to analyze workflow graphs. In *Advanced information systems engineering*. Lecture notes in computer science (Vol. 2348, pp. 535–552). Berlin: Springer.

Weber, B., & Reichert, M. (2008). Refactoring process models in large process repositories. In *Advanced information systems engineering*. Lecture notes in computer science (Vol. 5074, pp. 124–139). Berlin: Springer.

Matthias Schrepfer is a Senior Business Process Manager and Business Process Consultant in the Business Excellence team at Zalando SE. He drives initiatives to ensure reliable and superior performance of Zalando's operational business processes. Matthias is leading cross-departmental projects that apply continuous improvement methodologies to sustain and optimize business process performance. Besides that, he develops and adapts methodologies tailored towards specific goal and project scenarios within Zalando. Prior to his focus on operational excellence, he developed products and methodologies for business process monitoring at Zalando. Matthias implemented and established tools that monitor the process performance of core business processes in real time. This included the implementation of alerting capabilities and troubleshooting features to enable immediate impact detection and resolution. All tools were tailored for both business as well as technical teams. Matthias enabled these teams to effectively control and manage their business processes.

Matthias Kunze is a business process manager and business process consultant in the Business Excellence team at Zalando SE. The team is ensuring the reliable performance of Zalando's operations and business continuity, and enables stakeholders within Zalando to effectively understand, control, and manage their business processes. Before joining Zalando in 2015, he was a student, research associate, and postdoctoral researcher at the Hasso Plattner Institute in Potsdam. Besides his contributions to many practical aspects of BPM, his research work focussed on the behavioural analysis of business process models. Matthias Kunze received his PhD in 2013 for his dissertation on "Searching Business Process Models by Example". Together with Mathias Weske he published the textbook "Behavioural Models—From Modelling Finite Automata to Analysing Business Processes" in 2016.

Gunnar Obst is a passionated p-manager—all about processes, projects and (software) product development. After passing his diploma as business engineer he worked now for over 15 years in positions, where he connected business- and technology-stakeholders. At Zalando he worked over 6 years in the technology department heading different positions. Starting as responsible for the standard enterprise-resource-planning system Zalando used that days. After building a team of product-, process- and support-manager around the system, he drove the software-development of the own created ERP system ZEOS (Zalando E-Commerce-Operating-System)—full responsible for the wholesale part. Bringing the system live and handed it over to product management, he started to create a central team for process management within the technology department. This includes building up a process monitoring and anomaly detection system. After introducing "Radical Agility" in Zalando Tech, he worked as Delivery Lead Platform Excellence to ensure, the Fashion Platform work as expected.

Juliane Siegeris is a full professor in the school of computing, communication and business of the Hochschule für Technik und Wirtschaft (University of Applied Science) Berlin, Germany. She is head of the program computer science and business administration. Her main research interests are in business process modeling with a focus on correctness criteria. After her PhD "A Methodology for Workflow Modeling—from business process modeling towards sound workflow specification" she worked as business analyst at the Fraunhofer Institute for Software and System Engineering and as process manager at gematik. There she led the installment of a process map for the new process organization.