Research Center
Finance & Information Management

Fraunhofer FIT
Project Group
Business & Information
Systems Engineering

PRISM – A Predictive Risk Monitoring Approach for Business Processes

by

Raffaele Conforti[1], Sven Fink[2], Jonas Manderscheid, Maximilian Röglinger

in: Proceedings of the 14th International Conference on Business Process Management (BPM), Rio de Janeiro, Brazil, September 2016, p. 383-400

[1] Post-Doc, QUT
[2] Senacor Technologies AG

University of Augsburg, D-86135 Augsburg
Visitors: Universitätsstr. 12, 86159 Augsburg
Phone: +49 821 598-4801 (Fax: -4899)

University of Bayreuth, D-95440 Bayreuth
Visitors: Wittelsbacherring 10, 95444 Bayreuth
Phone: +49 921 55-4710 (Fax: -844710)

www.fim-rc.de

PRISM – A Predictive Risk Monitoring Approach for Business Processes

Raffaele Conforti[1], Sven Fink[2], Jonas Manderscheid[2], Maximilian Röglinger[3]

[1]Queensland University of Technology, Brisbane, Australia
raffaele.conforti@qut.edu.au
[2]FIM Research Center, University of Augsburg, Augsburg, Germany
sven.fink@student.uni-augsburg.de; jonas.manderscheid@fim-rc.de
[3]FIM Research Center, University of Bayreuth, Bayreuth, Germany
maximilian.roeglinger@fim-rc.de

Abstract. Nowadays, organizations face severe operational risks when executing their business processes. Some reasons are the ever more complex and dynamic business environment as well as the organic nature of business processes. Taking a risk perspective on the business process management (BPM) lifecycle has thus been recognized as an essential research stream. Despite profound knowledge on risk-aware BPM with a focus on process design, existing approaches for real-time risk monitoring treat instances as isolated when detecting risks. They do not propagate risk information to other instances in order to support early risk detection. To address this gap, we propose an approach for predictive risk monitoring (PRISM). This approach automatically propagates risk information, which has been detected via risk sensors, across similar running instances of the same process in real-time. We demonstrate PRISM's capability of predictive risk monitoring by applying it in the context of a real-world scenario.

Keywords: Business Process Management, Risk-aware BPM, Risk Propagation, Predictive Risk Monitoring

1 Introduction

The pressing need for organizations to increase productivity, to achieve operational excellence, and to save costs has been and still is one of the driving forces for the adoption of business process management (BPM) methods and technologies [1]. Due to an increasingly complex and dynamic business environment as well as the organic nature of processes, organizations are exposed to severe operational risks (e.g., the violation of the four-eye principle or a payment default of a customer engaged in multiple instances) when executing their business processes [2, 3].

In the attempt of solving this problem, industry and academia have proposed several solutions. From an industry perspective, there are legislative initiatives such as Basel II [4] and Sarbanes-Oxley Act [5]. In the academic world, previous research recognized the importance of incorporating a risk perspective in all activities of the BPM lifecycle

[3, 6, 7]. A detailed analysis of research conducted in the area of risk-aware BPM is presented by Suriadi et al. [8]. Accordingly, the effort of the academic world shows a bias toward the process design phase of the BPM lifecycle [9]. Beyond risk-aware process design, there are also works that take a risk perspective when valuating and comparing process models [10, 11]. Therefore, Suriadi et al. [8] particularly highlight the need for research on risk-informed process execution. This need is supported by Recker and Mendling [12], who point to a lack of research on real-time process monitoring and controlling. Although recent studies attempt to address this gap regarding risk-informed process execution and monitoring via approaches for real-time risk or deviance monitoring [13, 14], risk mitigation [15], and the avoidance of risk during runtime [16], the timely detection of process risks still is an open challenge.

Current approaches to real-time risk detection monitor running instances via sensors [13]. Such sensors operate at the level of process instances, using information about running instances and log data from completed process instances. Although risk eventuation in process instances is not necessarily affected by other running instances, external factors (e.g., customer behavior, characteristics of process inputs) play an influential role for the eventuation of risks. Due to such factors, risk monitoring at process instance-level may not be sufficient when instances are considered in isolation [17]. To the best of our knowledge, there are no approaches that share risk information across multiple process instances for the predictive monitoring and early detection of risks.

Against this background, we propose an approach for predictive risk monitoring (PRISM), which builds on and extends the work of Conforti et al. [13]. The PRISM approach aims at supporting early risk detection by automatically propagating risk information, which has been detected by sensors, across similar running instances of the same process in real-time. To do so, the PRISM approach uses a similarity-weighted process instance graph (PING) and a risk propagation algorithm.

The remainder of this paper is organized as follows: Section 2 discusses related work in the areas of risk-aware BPM and process similarity. Section 3 presents the PRISM approach, elaborating on the PING and the risk propagation algorithm. Section 4 illustrates PRISM's effectiveness when used in the context of a real-life scenario. Section 5 concludes the paper, discusses limitations and future work.

2 Theoretical Background and Preliminaries

In this section, we compile theoretical background on risk-aware BPM and on previous work related to process similarity.

2.1 Risk-aware Business Process Management

Risk management techniques found their way into many different fields. At the strategic level, risk management standards prescribe general guidelines for identifying, analyzing, evaluating, and handling risks [18, 19]. Though being of great importance, such guidelines are mostly meta-models, sketchy, and fail to provide specific guidance on how to operationalize risk management strategies in business processes. Conforti et al.

[13, 15, 16] thus proposed to enrich the traditional BPM lifecycle [9] with elements of risk management. This enables the four phases of the BPM lifecycle, i.e., *process design*, *process implementation*, *process enactment*, and *process analysis*, to become risk-aware. The resulting risk-aware BPM lifecycle is shown in Fig. 1.

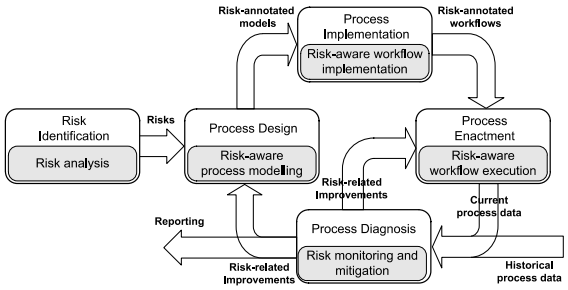


Fig. 1. Risk-aware BPM lifecycle [16]

The risk-aware BPM lifecycle starts with the *risk identification* phase. In this phase, risks that may eventuate during the execution of a business process are identified. The output of this phase is a set of risks, expressed as risk conditions, which are then mapped to process model-specific aspects in the *process design* phase. This mapping results in risk-annotated process models. In the *process implementation* phase, a more detailed assignment of risks and faults to specific aspects of a process model, e.g., the content of data variables and resource states, is conducted. A risk-aware process engine then executes this process model in the *process enactment* phase. Finally, based on the input of current and historical process data, risk conditions are analyzed in the *process diagnosis* phase, leading to risk-related improvements.

Considering the risk-aware BPM lifecycle, most approaches proposed for risk-aware BPM fall into the design phase. Among these approaches, we can distinguish between approaches that focus on the analysis and modeling of process risks via new risk constructs [3, 20–22] or based on the use of existing risk analysis methods [6, 23–25]. We refer to the work of Suriadi et al. [8] for a comprehensive discussion. Regarding the process diagnosis phase, we find the works of Pika et al. [26] and Suriadi et al. [27], who analyze process data to retrieve risk information. Pika et al. [26] propose an approach that uses statistical analysis to predict overtime risks, whereas Suriadi et al. [27] use classification algorithms to conduct a root cause analysis of risks.

In light of the need for research on risk-informed process execution/enactment [8], it is important to focus on risks that can be identified within the boundaries of a process. Thereby, a process risk is the chance of something happening that will impact the objectives of a process and is measured in terms of likelihood and consequence [28]. The work of Conforti et al. [13] focuses on real-time risk detection. The approach, which is based on sensors, detects risks via real-time monitoring of risk conditions. Though being capable of monitoring a process instance using current and historical information, sensors consider process instances as independent. This limits the capabilities of the approach since it is unable to detect the eventuation of process risks based on information about process risks that eventuate in other instances. Nonetheless, it offers a good starting point for addressing the problem.

2.2 Similarity Measures in BPM

For the purpose of the PRISM approach, we compare process instances in order to determine whether and how strongly a risk, which has been detected in one instance, influences other running instances of the same process. To compare process instances, it is necessary to measure the similarity of instances. In the literature, several approaches have been proposed and, in the area of BPM, we must distinguish between measuring similarities among process models [29–33] and process logs [34].

Similarities among process models can be categorized in structural similarities [29] and behavioral similarities [30, 31]. Approaches referring to structural similarities compare two process models at structural level. This is achieved by determining the number of structural changes required (e.g., flows, gateways, and tasks) for two process models to match. Approaches that deal with the behavioral similarity of process models require more advanced techniques. Two process models are compared regarding the set of possible executions that can be generated using these models. A similar approach is used by similarity measurement that operates on process logs [35], while in this case the set of possible executions is already contained in the log. For these forms of similarity, two characteristics need to be kept in mind: first, instances belonging to the same process model make structural similarity pointless and, second, multiple completed instances are required in order to reasonably compare logs.

As an approach for measuring the similarity of process logs, Song et al. [34] rely on trace profiles. Trace profiles are vectors, containing several items that describe the trace from a specific perspective (e.g., case attributes or involved tasks). Trace profiles build on historical data from process logs in order to obtain their information. In light of their multi-perspective vectorial representation, trace profiles can be easily compared using string similarity techniques [36, 37]. Song et al. [34] show how a similarity measure based on trace profiles enhances discovering process models. This is why the PRISM approach builds on the work of Song et al. [34].

3 The Predictive Risk Monitoring Approach

We now present the PRISM approach that builds on and extends the work of Conforti et al. [13]. The approach encompasses a similarity-weighted PING and a risk propagation algorithm. For the sake of completeness, we first sketch the approach of Conforti et al. [13].

3.1 The Sensor-based Approach to Risk Detection

In the sensor-based approach of Conforti et al. [13], a fault is an undesired state of a process (e.g., a process violating a service level agreement). In order to minimize the negative effects of faults, it is important to detect the risk of a fault as early as possible. Conforti et al. [13] achieve this through the use of digital risk sensors. However, the approach would also be suitable in the case of physical sensors [38].

Sensors are defined at design time on top of an executable process model. Each sensor is associated with a risk condition that captures the situation in which the risk related

to a distinct fault may occur. A risk condition combines a risk likelihood (henceforth referred to as risk $r_i(t)$ in instance i at a given point in time t), i.e., the probability of the fault to occur, and a threshold TRE, i.e., a risk value that an organization is willing to tolerate. As process models can contain several sensors referring to different faults with individual risk conditions, each sensor must be treated separately.

At execution time, when a new instance is created, the sensors associated with the process model are enabled. The sensors monitor the process instance by evaluating the associated risk condition. A risk condition is evaluated either based on a given sampling rate or on the occurrence of a specific event by looking into historical and current process execution data. Finally, whenever a risk is detected the system automatically triggers a notification to the process administrator, who will act accordingly.

The PRISM approach builds on top of and extends the work of Conforti et al. [13]. It is based on the idea that similar process instances feature a similar risk exposure. We thus assume that identical instances have the same risk exposure. As other approaches (e.g., case-based reasoning or adaptive case management) rely on similarity measures to determine similar instances [39, 40], we adopted similarity as a proxy to estimate the risk exposure of other currently running instances. On this assumption, we use the sensor-based detection of a risk in a distinct instance as a trigger for checking whether the risk is likely to eventuate in similar process instances, too. This is achieved by propagating information about the detected risk from the process instance for which the risk has been detected to other currently running instances. To determine to which instances a detected risk should be propagated and how strongly the related effect should be, the PRISM approach compares different instances regarding their similarity. The risk propagation triggers a manual evaluation of the corresponding sensor from the receiving instance, taking the propagated risk as well as the similarity between the source and the receiving instance as input for evaluating the risk condition.

Fig. 2 illustrates the idea behind the PRISM approach for two running instances using a single sensor as example. In this example, the sensor is monitoring an *unfulfillment risk*. This sensor relates to a situation where a process instance executes a distinct task too many times, with the related risk condition checking for loops. We assume that both instances here have a high risk exposure and are similar. In the following, we refer to instance 1 as source. Accordingly, the source's unfulfillment risk sensor detects a risk that exceeds the given threshold at time $t = 2$. This calculation is based on available historical data (i.e., already executed log traces), enabling to analyze past executions of the process in focus. In Fig. 2, the bold dashed line from the source to instance 2 visualizes the risk propagation, triggered by the detection of the unfulfillment risk. The propagation leads to rechecking the risk conditions in instance 2. In our example, the unfulfillment risk is detected in instance 2 triggered by the risk propagation. PRISM therefore enables detecting the unfulfillment risk in instance 2 earlier, i.e., in $t = 2$ instead of $t = 3$, than the sensors of instance 2 would have done without risk propagation.

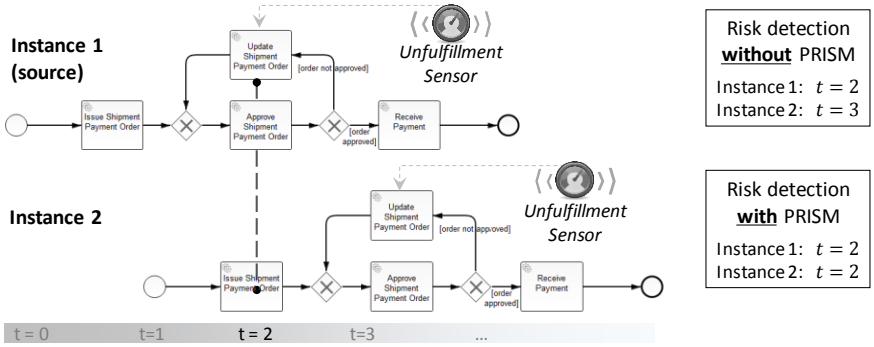


Fig. 2. Example of time advantage through the PRISM approach with two instances

As there usually is more than one running instance of the same process, the PRISM approach propagates risk information among these instances to enable early risk detection. In order to perform the risk propagation, the PRISM approach uses a similarity-weighted PING and a risk propagation algorithm whenever a sensor detects a risk. We introduce both concepts below.

3.2 Similarity-weighted Process Instance Graph

To propagate risk information among running instances of a process, we rely on a similarity-weighted PING. The PING virtually links instances using their similarity as edge weights. The PING can be interpreted as a temporal snapshot of all process instances, which we use to determine whether and how strongly a risk detected in the source instance influences other instances of the same process.

Formally, the PING is a graph $PING = (V, E)$, where $V = (1, \dots, n)$ is the index set of all running instances with index 1 representing the source, i.e., the instance that triggers the creation of the PING. Further, $E(t) \in \mathbb{R}^{n \times n}$ is the triangular adjacency matrix that captures the similarity $s_{i,j}(t)$ of two instances at a distinct point in time. The adjacency matrix relates to a distinct point in time as the similarity of instances may change over time when their execution is progressing. Each time a PING is created, the process instances receive a new index. As running instances terminate and new instances begin, $E(t)$'s dimensionality changes constantly. By assigning new indexes, we ensure that, for a distinct point in time, only running instances are considered and that no unnecessarily large data structures must be handled in real-time.

$$E(t) = \begin{pmatrix} s_{1,1}(t) & \cdots & s_{1,n}(t) \\ \vdots & \ddots & \vdots \\ s_{n,1}(t) & \cdots & s_{n,n}(t) \end{pmatrix}, s_{i,j}(t) \in [0,1] \quad (1)$$

The adjacency matrix $E(t)$ is symmetric except for those elements that contain the source, i.e., $s_{i,j}(t) = s_{j,i}(t)\ \forall i, j \in V \setminus \{1\}$. The source instance only propagates risk information, i.e., $s_{i,1}(t) = 0$. The source needs not receive any risk information as one of its sensors has initially detected the risk that triggered the creation of the PING. For the same reason, all other instances do not propagate risk information to themselves,

i.e., $s_{i,i}(t) = 0$. In all other cases, $s_{i,j} = 0$ if two instances are absolutely different and $s_{i,j} = 1$ if the instances are perfectly equal according the similarity measure.

In the PRISM approach, we calculate the similarity of instances in line with Song et al. [34], i.e., based on trace profiles (section 2.2). We build trace profiles based on explicit information (e.g., names of tasks) and on derived information (e.g., number of events in a trace). Each instance can be characterized by multiple profiles. A profile is an n-dimensional vector where n indicates the number of items extracted from a log. A profile $c_{p,i}$ refers to a vector $\langle a_{i,1}, a_{i,2}, \dots, a_{i,n} \rangle$, where $a_{i,k}$ denotes the amount of item k's appearances in instance i for profile p. For each profile p, the similarity of two instances $s_{p,i,j}(t)$ is calculated as shown in Equation (2).

$$s_{p,i,j}(t) = 1 - \frac{d\left(c_{p,i}(t), c_{p,j}(t)\right) - d_{\min}}{d_{\max}(t) - d_{\min}} = 1 - \frac{\sqrt{\sum_{k=1}^{n} |a_{i,k}(t) - a_{j,k}(t)|^2}}{\max_{j \in V} \left\{ \sqrt{\sum_{k=1}^{n} |a_{i,k}(t) - a_{j,k}(t)|^2} \right\}} \quad (2)$$

To determine the similarity of instances i and j for profile p, the respective normalized distance is subtracted from 1. We assume $d_{\min}$ to be 0 as instances can be identical. As an increasing value is needed to capture more similarity, we subtract the normalized distance from 1. In order to calculate the distance, it is possible to apply different distance measures (e.g., Euclidean, Hamming, or Jaccard) as shown by Song et al. [34]. We decided in favour of the Euclidean distance, which led to good results when used for trace clustering [34]. As the focus of this paper is not on the identification of the best similarity measures, other distance could have been used as well. We will get back to this issue in the critical reflection. To normalize the distance between two instances, an operation necessary to compare the distance between any pair of instances, we divide the distance of the respective trace vectors by the maximum distance available.

To derive a single value that represents the similarity of two instances across all trace profiles in focus, we determine an overall similarity by calculating the weighted average of all profiles that relate to the involved instances. We thus integrate the similarity of all profiles based on their relative importance for the estimation of risks. In Equation (3), w_p represent the weights of all profiles p with $\sum_{p \in P} w_p = 1$.

$$s_{i,j}(t) = \sum_{p \in P} w_p \cdot s_{p,i,j}(t) \quad (3)$$

3.3 Risk Propagation Algorithm

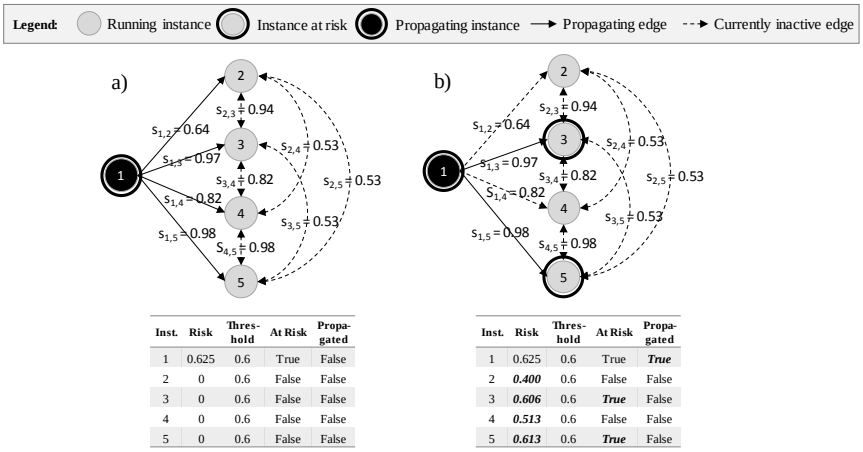
In case a sensor in a distinct instance detects a risk (i.e., the risk condition evaluates to true because the risk probability exceeds the threshold), the risk propagation algorithm cascades this information across all currently running instances. To do so, the risk propagation algorithm builds on the PING and estimates the eventuation probability (i.e., the probability that the risk condition of the other instances also evaluates to true) of the detected risk in other instances using similarities, inspired by the signal/collect programming model [41]. The risk propagation algorithm follows a two-phase approach, i.e., initial propagation and re-propagation. If a propagation is successful, we refer to the state of the respective instance as "at risk."

In the initial propagation phase, the source propagates the detected risk (i.e., a risk likelihood that exceeds a given threshold) to all other instances (see black solid lines in

Fig. 3 a). The propagation accounts for the source's similarity with other running instances. Acting on the assumption of a proportional relationship between similarity and risk exposure, we estimate the risk of a receiving instance $r_j(t)$ at a distinct point in time according to Equation (4).

$$r_j(t) = \begin{cases} s_{i,j}(t) \cdot r_i(t) & \text{if } s_{i,j}(t) \cdot r_i(t) > TRE \\ 0 & \text{else} \end{cases} \quad (4)$$

If the risk multiplied with the similarity of the propagating instance (i.e., the source instance in the initial propagation phase) and the receiving instance exceeds the threshold pre-defined for a sensor, the respective product is assigned to the receiving instance in terms of a signal/collect procedure. If the threshold is not exceeded, the product is 0. As instances typically are different and thus do not feature the same risk exposure, not all instances receive the same propagated risk. As we look at pairwise similarity, we do not cumulate received risk values in an instance during propagation, as this would result in an overestimation. In Fig. 3 and Fig. 4, the results of the initial propagation are written into the table below the graph (even if the threshold is not exceeded) to illustrate the risk received in an instance. Fig. 3 a shows the situation when the source detects risk and propagates the risk value to all other running instances (see black solid lines). As not for all instances the propagated risk (i.e., similarity times risk of the source) exceeds the threshold, only instances 3 and 5 reach the "at risk" state (Fig. 3 b).



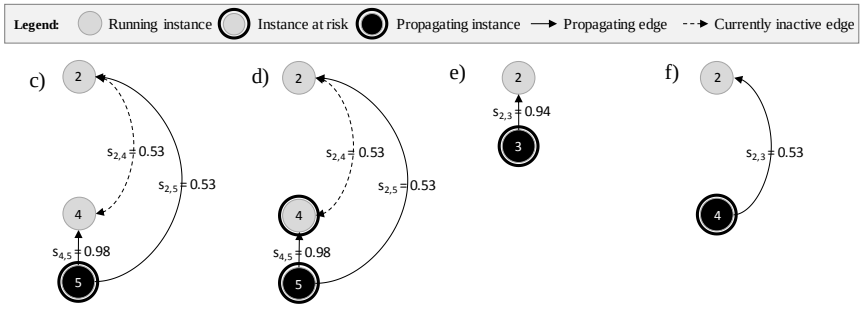
Inst.	Risk	Threshold	At Risk	Propagated
1	0.625	0.6	True	False
2	0	0.6	False	False
3	0	0.6	False	False
4	0	0.6	False	False
5	0	0.6	False	False

Inst.	Risk	Threshold	At Risk	Propagated
1	0.625	0.6	True	***True***
2	***0.400***	0.6	False	False
3	***0.606***	0.6	***True***	False
4	***0.513***	0.6	False	False
5	***0.613***	0.6	***True***	False

Fig. 3. Visualization of the initial propagation phase

For all instances whose product of risk and similarity exceeds the sensor's threshold, the iterative re-propagation phase is triggered (Fig. 4). This phase and the iterative character are necessary as a process instance can get into the "at risk" state by receiving risk transitively propagated from the source (e.g., the similarity between the source and instance 4 is rather low, but much higher between instances 5 and 4). As the risk is getting smaller with each re-propagation (i.e., $s_{ij} < 1$), we account for "paths" from the source to other nodes with a maximum length of 2. Using longer "paths" for risk propagation than assumed is possible and provides experts with the ability to customize the PRISM approach to their needs.

In the re-propagation phase, all instances that have been classified as "at risk" in the initial propagation or a previous re-propagation are sorted according to their risk exposure. We start with the instance that has the highest risk exposure. From the initial propagation (Fig. 3) to the re-propagation (Fig. 4), we can see that instance 5 with the highest received risk is considered first. The instance's received risk is used for the first re-propagation to all instances that have not yet been classified as "at risk". As, for example, the re-propagation from instance 5 to 4 exceeds the threshold of instance 4, instance 4 is classified as "at risk" after the re-propagation (Fig. 4 d). The sorting of the instances "at risk" is not mandatory. However, starting with the highest risk, the algorithm terminates faster. All instances that went into the "at risk" state due to the current re-propagation are then added to the set of instances to be considered in the next re-propagation (e.g., instance 4 after the re-propagation of instance 5). Instances that already re-propagated need not be considered further (e.g., instance 5). This iterative procedure continues until there are no more "at risk" instances that have not yet re-propagated or until the specified maximum number of re-propagations is reached.



Inst.	Risk	Thres-hold	At Risk	Propa-gated
1	0.625	0.6	True	True
2	0	0.6	False	False
3	0.606	0.6	True	False
4	0	0.6	False	False
5	0.613	0.6	True	False

Inst.	Risk	Thres-hold	At Risk	Propa-gated
1	0.625	0.6	True	True
2	***0.325***	0.6	False	False
3	0.606	0.6	True	False
4	***0.601***	0.6	***True***	False
5	0.613	0.6	True	***True***

Inst.	Risk	Thres-hold	At Risk	Propa-gated
1	0.625	0.6	True	True
2	***0.570***	0.6	False	False
3	0.606	0.6	True	***True***
4	0.601	0.6	True	False
5	0.613	0.6	True	True

Inst.	Risk	Thres-hold	At Risk	Propa-gated
1	0.625	0.6	True	True
2	***0.319***	0.6	False	False
3	0.606	0.6	True	True
4	0.601	0.6	True	***True***
5	0.613	0.6	True	True

Fig. 4. Visualization of the re-propagation phase

Although, in Fig. 4, instance 4 is added to the relevant instances for re-propagation, instance 3 still has the highest risk of the remaining "at risk" instances. Thus, the algorithm starts the second re-propagation with instance 3 (Fig. 4 e). As no further instance exceeds the threshold based on the re-propagation, the algorithm terminates after the re-propagation of instance 4 (Fig. 4 f). As result, the PRISM approach classifies four out of five instances as "at risk".

4 Demonstration

To demonstrate its effectiveness, we apply the PRISM approach to a process for a personal loan or overdraft application in the context of a real-world scenario from a Dutch financial institute. The corresponding log data was released as part of the BPI Challenge

held in conjunction with the 8th International Workshop on Business Process Intelligence 2012 [42]. As a prerequisite for our demonstration, we implemented the PRISM approach as an extension of the workflow management system Camunda.[1] Below, we first present the process model and data from process execution. After that, we outline how we operationalized the sensors and similarity measures. Finally, we compare the results of the PRISM approach with the sensor-based approach by Conforti et al. [13].

4.1 Demonstration Design and Data Set

The application process for a personal loan or overdraft (Fig. 5) starts with the submission of an application. The financial institute may already decline the application at this point in time, a decision that will bring the process to a quick end. The financial institute may pre-accept the application for further processing. In this case, one of the financial institute's employees first adds missing information until the application is completed. The employee then selects and creates an offer, sends the offer to the customer, and adds this information to the application. After that, the employee calls the customer periodically. After the customer made her decision, the application will be finally assessed while adding still missing information.

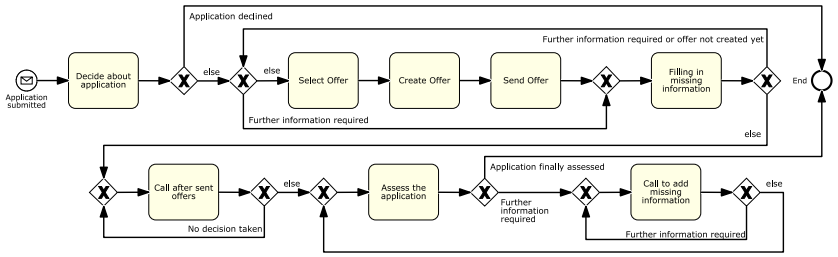


Fig. 5. Filtered process model of the personal loan and overdraft application process

The corresponding log contains traces with events that cover a period of six months, i.e., from October 2011 to March 2012. Each line of the log corresponds to an executed task that relates to a distinct instance. The log also includes the resource that executed the task, the timestamp of task completion as well as the loan or overdraft amount requested by the customer. As an example, Table 1 shows the trace of the process instance with the CaseID 175585 with a requested amount of 22,000 EUR.

To improve its quality, we pre-processed the log via a two-phase filtering approach. In the first phase, we removed infrequent labels, applying the "Filter Log using Simple Heuristics" plugin of ProM with a boundary of 90%. In the second phase, we removed infrequent behavior from the log based on the approach by Conforti et al. [43]. The pre-processed log contained 11 unique tasks and 9,350 instances resulting in 91,500 events. Having pre-processed the log, we extracted the process model used for our demonstration (Fig. 5).

[1] http://www.camunda.com/. The authors are happy to provide the source code upon request.

Table 1. Log trace for CaseID 175585

#	Task	Resource	Complete_Timestamp
1	START	artificial	2011/10/08 14:50:02.113
2	A_SUBMITTED	112	2011/10/08 14:50:02.113
3	A_PARTLYSUBMITTED	112	2011/10/08 14:50:02.243
4	A_PREACCEPTED	112	2011/10/08 14:50:42.639
5	O_SELECTED	11000	2011/10/08 14:56:37.300
6	O_CREATED	11000	2011/10/08 14:56:39.224
7	O_SENT	11000	2011/10/08 14:56:39.271
8	W_Filling in information for the application	11000	2011/10/08 14:56:41.605
9	W_Calling after sent offers	11000	2011/10/08 14:57:16.346
…	W_Calling after sent offers	…	…
18	W_Calling after sent offers	11049	2011/10/24 12:20:18.377
19	W_Assessing the application	10629	2011/10/27 13:35:15.895
20	W_Calling to add missing information to the application	10939	2011/10/27 18:42:05.333
21	W_Assessing the application	10629	2011/10/28 08:38:08.642
22	END	Artificial	2011/10/28 08:38:08.643

In our demonstration, we measured how often the PRISM approach was capable of predicting the eventuation of a risk measured by a sensor. To substantiate the advantage gained by applying PRISM, we determined how long before the risk's eventuation the prediction was made. Additionally, we measured how often PRISM was unable to predict a risk detected by a sensor or produced an erroneous prediction where no risk has been detected before. To perform the demonstration and to ensure its replicability, we replayed the execution of process instances according to the log data.

4.2 Operationalization for PRISM Demonstration

Before starting the replay, we implemented one sensor in the application process. This sensor monitors the *unfulfillment risk* as introduced in Conforti et al. [13]. This risk occurs if an instance executes a task too often, a situation that occurs in loops. To avoid slowdowns and livelocks during execution, a task is assigned a maximum amount of executions per instance (MAE), which may be defined as part of an internal regulation or service level agreement. In our demonstration, we monitor the unfulfillment risk with respect to the "*W_Calling after sent offers*" task with $MAE = 10$. As the process log did not come with any additional information on a defined maximum time or maximum amount of executions, we had to estimate a sensible value. We expected that not more than 10% of the instances are faulty and set $MAE = 10$, as it represents the 92% quantile of the MAE distribution contained in the log. The instance from Table 1, for example, reached this amount as the task "*W_Calling after sent offers*" is executed 10 times. The unfulfillment risk sensor monitors the risk according to the risk condition shown in Equation (5), whenever an instance executes the "*W_Calling after sent offers*" task.

$$\min\left\{\frac{\#instances \geq MAE}{\#instances \geq current\ amount\ of\ task\ execution}; 1\right\} > TRE \tag{5}$$

Accordingly, we divide the amount of instances that executed the task at least as often as the defined MAE by the amount of instances that executed the task at least as often as the instance that triggered the sensor calculation. In case an instance already executed the task more often than the defined MAE, the left value of the risk condition must not exceed 1, as it reflects the probability that the instance exceeds the specified

maximum amount of executions. Whenever the probability exceeds the defined threshold TRE, the PING is created and the risk propagation algorithm is triggered. We chose $TRE = 0.6$ to capture a risk-neutral setting. The threshold can also be adapted to reflect more risk-averse or risk-seeking settings. As the sensor calculates the risk based on the already executed instances of a process, we had to ensure that a sufficient amount of instances has already been executed before the first risk propagation is triggered. We thus started 40 instances of the log in order not to perform the sensor's calculation on an empty database. This amount of instances shaped up as sufficient in some scenarios.

To derive the similarity values used for risk propagation, we used two profiles. The first profile builds on case attributes from the log, the second considers the amount of executed tasks of a distinct instance. However, the log only offers two attributes, i.e., the requested amount of money and resource executing a task, whereby the resource is missing for many tasks. We decided to calculate the distance vector of the first profile with just one element, i.e., the difference of the requested amounts of money. For the second profile, we used the amount of executions per task. For both profiles, the maximum distance vector was determined based on instances in the snapshot (e.g., instances that were currently running when the risk was detected, as explained for the PING in section 3.2). The maximum distance vector we choose for normalizing the distance values must not exceed the average distance from the snapshot by 80%. We decided for this assumption to minimize the effect of outliers, which would cause many false positives. To further reduce the number of false positive risk propagations, we limited the amount of instances taken into consideration for propagation. We reduced the running instances by those who have already passed the task our sensor is attached to. Instances that proceeded to the task *"W_Assessing the application"* cannot be faulty anymore. At least the sensor related to their instance would have had to indicate potential risk.

The calculation of both profiles is performed according to Equation (2). We take the profiles' result and their weights in order to derive a similarity for two distinct instances (Equation 3). As the log contained almost no case attributes, we selected the task profile as the leading profile. We thus assigned a weight of 0.6 to the task profile and 0.4 to the attribute profile as shown in Equation (6).

$$s_{i,j}(t) = \sum_{p \in P} w_p \cdot s_{p,i,j}(t) = 0.4 \cdot s_{\text{attribute},i,j}(t) + 0.6 \cdot s_{\text{task},i,j}(t) \quad (6)$$

4.3 Results of the Replay and Discussion

To show a successful implementation and validate the PRISM approach, we selected instances that started between October 1st and 11th 2011. This set of instances captures an average workload per week from the process log, including a sufficiently large number of instances to train the PRISM approach. As declined loan requests would lead to negligible similarity values and receive no risk propagation, we only looked at instances with accepted loan requests. With the replay of the resulting 241 instances, we gained the results as illustrated in Table 2.

In this setting, the PRISM approach predicted with an accuracy of 86.72% (209 out of 241 instances). In the used log data, the sensor detected a risk for 14 instances. Out of these instances, 13 where correctly identified as being "at risk" before the respective

instances' own sensors detected the risk. The instance with the missing prediction appeared due to propagation algorithm. The algorithm triggers the risk propagation upon the detection of risk in a sensor. Thus, the first instance that runs into a risk and triggers the first propagation cannot receive any information from an earlier propagation.

When we look into details, the time saved by the PRISM approaches averages 4 days 18 hours compared to a risk detection without risk propagation among similar instances. The average execution time of the covered 241 instances amounted to 65 days 12 hours. For our example trace (i.e., CaseID 175585) in Table 1, the unfulfillment risk was identified in task 16 with the 8th execution of task "*W_Calling after sent offers*" and caused a time advantage of 5 days 20 hours.

Table 2. Contingency table for predicting risk with PRISM

	Sensor detected risk	**Sensor detected no risk**
PRISM risk predicted	13/14 = 92.86%	31/227 = 13.66%
PRISM no risk predicted	1/14 = 7.14%	196/227 = 86.34%

Finally, we critically reflect on the results of the demonstration, as we made some assumptions when operationalizing the PRISM approach (e.g., similarity measure, normalization of the distance vectors). The different profiles of a process allow for different perspectives on similarity and provide high flexibility. The process log we used for the replay, however, only contained very few attributes we could use for building profiles. Thus, it needs to be checked how the availability of more attributes influences the demonstration results. Further, the relation of missing to false predictions is influenced by the risk conditions, thresholds, and the maximum number of re-propagations. These properties can be adapted according to a process manager's risk attitude. Nevertheless, we were able to demonstrate the effectiveness of the PRISM approach as we gained good results based on limited information. In addition, we only analyzed instances that started in a limited time period. Although this set of instances represented an average work week, it covers only a subset of the log data. We deliberately restricted the demonstration to a smaller subset to better understand what is happing during risk propagation. Thus, a next step would be to further develop the prototype and to apply the PRISM approach to the entire log.

5 Conclusion and Critical Discussion

In this paper, we proposed the predictive risk monitoring (PRISM) approach that automatically propagates risk information, detected by risk sensors, across similar instances of the same process in real-time. On the assumption that similar process instances have a similar risk exposure, the PRISM approach uses a similarity-weighted process instance graph (PING) that can be interpreted as a snapshot of all currently running instances. The PING virtually links all currently running instances and uses the similarity of these instances as edge weights. Based on the PING, a risk propagation algorithm then determines whether and how strongly a detected risk influences other instances. The PRISM approach intends to detect risks earlier than approaches without risk propagation. In the context of a real-world scenario, we demonstrated that the similarity

assumption holds true and that the PRISM approach is indeed able to detect risks earlier than the approach of Conforti et al. [13].

Although we were able to demonstrate the effectiveness of the PRISM approach and the feasibility of the underlying assumptions based on real-world data, the approach is beset with limitations that stimulate future research. First, the PRISM approach is based on a distinct similarity measure as well as on the assumption of a proportional relation between similarity and risk exposure. Future research should analyze whether other similarity measures and other relation types help achieve better risk prediction results. Second, it is time-consuming to parameterize the PRISM approach. Currently, the parameterization needs to be strongly geared toward the properties of the process log at hand. Future research should explore into methods that help parameterize the PRISM approach. Third, the information we use as input for the PRISM approach grounds on risk information triggered by risk sensors. We do not consider other input than log data. It might be useful to account for information from outside the process such as the context in which the process is executed (e.g., market fluctuations) or organizational risks (e.g., dependencies on third parties) to enhance predictive risk detection. Ideas may be derived from risk monitoring approaches applied in other domains as well as from more sophisticated propagation algorithms (e.g., belief propagation). This can help overcome current shortcomings of the PRISM approach (e.g., the re-propagation sequence and the termination rule). Likewise, the PRISM approach would benefit from further evaluation by means of sensitivity analyses, robustness tests, and case studies. Case studies would also help gain experience with estimating the needed parameters.

Acknowledgements

This research is partially funded by the ARC Discovery Project DP150103356 and was partially carried out in the context of the Project Group Business and Information Systems Engineering of the Fraunhofer Institute for Applied Information Technology FIT.

References

1. van der Aalst, W.M.P.: Business Process Management: A Comprehensive Survey. ISRN Softw. Eng. 2013, 1–37 (2013).
2. Beverungen, D.: Exploring the Interplay of the Design and Emergence of Business Processes as Organizational Routines. Bus. Inf. Syst. Eng. 6, 191–202 (2014).
3. zur Muehlen, M., Rosemann, M.: Integrating Risks in Business Process Models. In: 16th Australasian Conference on Information Systems. pp. 62–72. Association of Information Systems (2005).
4. Basel Committee on Banking Supervision: Basel II: International Convergence of Capital Measurement and Capital Standards. (2006).
5. Oxley, M.G., Sarbanes, P.: Sarbanes Oxley Act of 2002. 745–810 (2002).
6. Mock, R., Corvo, M.: Risk analysis of information systems by event process chains. Int. J. Crit. Infrastructures. 1, 247–257 (2005).
7. Betz, S., Hickl, S., Oberweis, A.: Risk-Aware Business Process Modeling and Simulation Using XML Nets. In: 13th Conference on Commerce and Enterprise Computing. pp. 349–356. IEEE (2011).
8. Suriadi, S., Weiß, B., Winkelmann, A., ter Hofstede, A.H.M., Adams, M., Conforti, R., Fidge, C.J., La Rosa, M., Ouyang, C., Pika, A., Rosemann, M., Wynn, M.: Current Research

in Risk-aware Business Process Management — Overview, Comparison, and Gap Analysis. Commun. Assoc. Inf. Syst. 34, 933–984 (2014).
9. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer Berlin Heidelberg, Berlin, Heidelberg (2013).
10. Bolsinger, M.: Bringing value-based Business Process Management to the Operational Process Level. Inf. Syst. E-bus. Manag. 13, 355–398 (2015).
11. Buhl, H.U., Röglinger, M., Stöckl, S., Braunwarth, K.S.: Value Orientation in Process Management. Bus. Inf. Syst. Eng. 3, 163–172 (2011).
12. Recker, J., Mendling, J.: The State of the Art of Business Process Management Research as Published in the BPM Conference. Bus. Inf. Syst. Eng. 58, 55–72 (2016).
13. Conforti, R., La Rosa, M., Fortino, G., ter Hofstede, A.H.M., Recker, J., Adams, M.: Real-Time Risk Monitoring in Business Processes: A Sensor-based Approach. J. Syst. Softw. 86, 2939–2965 (2013).
14. Manderscheid, J., Reißner, D., Röglinger, M.: Inspection Coming Due! How to Determine the Service Interval of Your Processes! In: Motahari-Nezhad, H.R., Recker, J., and Weidlich, M. (eds.) Business Process Management, LNCS, vol. 9253. pp. 19–34. Springer Berlin Heidelberg, Berlin, Heidelberg (2015).
15. Conforti, R., ter Hofstede, A.H.M., La Rosa, M., Adams, M.: Automated Risk Mitigation in Business Processes. In: Meersman, R., Panetto, H., Dillon, T., Rinderle-Ma, S., Dadam, P., Zhou, X., Pearson, S., Ferscha, A., Sonia Bergamaschi, and Cruz, I.F. (eds.) On the Move to Meaningful Internet Systems: OTM 2012, LNCS, vol. 7565. pp. 212–231. Springer Berlin Heidelberg, Berlin, Heidelberg (2012).
16. Conforti, R., de Leoni, M., La Rosa, M., van der Aalst, W.M.P., ter Hofstede, A.H.M.: A recommendation system for predicting risks across multiple business process instances. Decis. Support Syst. 69, 1–19 (2015).
17. Krumeich, J., Werth, D., Loos, P.: Prescriptive Control of Business Processes. Bus. Inf. Syst. Eng. 7, 1–40 (2015).
18. Association Information Systems Audit and Control: COBIT 5: A Business Framework for the Governance and Management of Enterprise IT. (2013).
19. AXELOS: Information Technology Infrastructure Library, https://www.axelos.com/best-practice-solutions/itil.
20. Jakoubi, S., Goluch, G., Tjoa, S., Quirchmayr, G.: Deriving Resource Requirements Applying Risk-Aware Business Process Modeling and Simulation. In: 16th European Conference on Information Systems. pp. 1542–1554. AIS (2008).
21. Sienou, A., Karduck, A.P., Lamine, E., Pingaud, H.: Business Process and Risk Models Enrichment: Considerations for Business Intelligence. In: 2008 IEEE International Conference on e-Business Engineering. pp. 732–735. IEEE (2008).
22. Singh, P., Gelgi, F., Davulcu, H., Yau, S.S., Mukhopadhyay, S.: A Risk Reduction Framework for Dynamic Workflows. In: 2008 IEEE International Conference on Services Computing. pp. 381–388. IEEE (2008).
23. Rotaru, K., Wilkin, C., Churilov, L., Neiger, D., Ceglowski, A.: Formalizing process-based risk with Value-Focused Process Engineering. Inf. Syst. E-bus. Manag. 9, 447–474 (2011).
24. Karagiannis, D., Mylopoulos, J., Schwab, M.: Business Process-Based Regulation Compliance: The Case of the Sarbanes-Oxley Act. In: 15th IEEE International Requirements Engineering Conference. pp. 315–321. IEEE (2007).
25. Lambert, J.H., Jennings, R.K., Joshi, N.N.: Integration of risk identification with business process models. Syst. Eng. 9, 187–198 (2006).
26. Pika, A., van der Aalst, W.M.P., Fidge, C.J., ter Hofstede, A.H.M., Wynn, M.T.: Predicting Deadline Transgressions Using Event Logs. In: Rosa, M. La and Soffer, P. (eds.) Business Process Management Workshops, LNBIP, vol. 132. pp. 211–216. Springer Berlin Heidelberg (2012).

27. Suriadi, S., Ouyang, C., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Root Cause Analysis with Enriched Process Logs. In: Rosa, M. La and Soffer, P. (eds.) LNBIP, vol. 132. pp. 174–186. Springer Berlin Heidelberg (2013).
28. Standards Australia and Standards New Zealand: ISO 31000:2009, Risk management — Principles and guidelines. (2009).
29. van Dongen, B.F., Dijkman, R.M., Mendling, J.: Measuring Similarity between Business Process Models. In: Bellahsène, Z. and Léonard, M. (eds.) Advanced Information Systems Engineering, LNCS, vol. 50474. pp. 450–464. Springer Berlin Heidelberg, Berlin, Heidelberg (2008).
30. Armas-Cervantes, A., Baldan, P., Dumas, M., García-Bañuelos, L.: Diagnosing Behavioral Differences Between Business Process Models: An Approach Based on Event Structures. Inf. Syst. 56, 304–325 (2016).
31. Polyvyanyy, A., Weidlich, M., Weske, M.: Isotactics as a Foundation for Alignment and Abstraction of Behavioral Models. In: Barros, A., Gal, A., and Kindler, E. (eds.) Business Process Management, LNCS, vol. 7481. pp. 335–351. Springer Berlin Heidelberg, Berlin, Heidelberg (2012).
32. Dijkman, R.M., Dumas, M., van Dongen, B.F., Käärik, R., Mendling, J.: Similarity of Business Process Models: Metrics and Evaluation. Inf. Syst. 36, 498–516 (2011).
33. Beheshti, S.-M.-R., Benatallah, B., Sakr, S., Grigori, D., Motahari-Nezhad, H.R., Barukh, M.C., Gater, A., Ryu, S.H.: Process Analytics - Concepts and Techniques for Querying and Analyzing Process Data. Springer International Publishing (2016).
34. Song, M., Günther, C.W., van der Aalst, W.M.P.: Trace Clustering in Process Mining. In: Ardagna, D., Mecella, M., and Yang, J. (eds.) Business Process Management Workshops, LNBIP, vol. 17. pp. 109–120. Springer Berlin Heidelberg, Berlin, Heidelberg (2009).
35. van Beest, N.R.T.P., Dumas, M., García-Bañuelos, L., La Rosa, M.: Log Delta Analysis: Interpretable Differencing of Business Process Event Logs. In: Motahari-Nezhad, H.R., Recker, J., and Weidlich, M. (eds.) Business Process Management, LNCS, vol. 9253. pp. 386–405. Springer Berlin Heidelberg, Berlin, Heidelberg (2015).
36. Hamming, R.W.: Error Detecting and Error Correcting Codes. Bell Syst. Tech. J. 29, 147–160 (1950).
37. Jaccard, P.: The Distribution of The Flora in The Alpine Zone. New Phytol. 11, 37–50 (1912).
38. Daniel, F., Eriksson, J., Finne, N., Fuchs, H., Karnouskos, S., Montero, P.M., Mottola, L., Oertel, N., Oppermann, F.J., Picco, G. Pietro, Römer, K., Spieß, P., Tranquillini, S., Voigt, T.: makeSense : Real-world Business Processes through Wireless Sensor Networks. In: 4th International Workshop on Networks of Cooperating Objects for Smart Cities, CONET/UBICITEC. pp. 58–72 (2013).
39. Minor, M., Bergmann, R., Görg, S.: Case-based adaptation of workflows. Inf. Syst. 40, 142–152 (2014).
40. Motahari-Nezhad, H.R., Bartolini, C.: Next Best Step and Expert Recommendation for Collaborative Processes in IT Service Management. In: Rinderle-Ma, S., Toumani, F., and Wolf, K. (eds.) Business Process Management, LNCS, vol. 6896. pp. 50–61. Springer Berlin Heidelberg, Berlin, Heidelberg (2011).
41. Stutz, P., Bernstein, A., Cohen, W.: Signal/Collect: Graph Algorithms for the (Semantic) Web. In: Patel-Schneider, P.F., Pan, Y., Pascal Hitzler, Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., and Glimm, B. (eds.) The Semantic Web – ISWC 2010, LNCS, vol. 6496. pp. 764–780. Springer Berlin Heidelberg, Berlin, Heidelberg (2010).
42. van Dongen, B.F.: BPI Challenge 2012.xes.gz, http://data.3tu.nl/repository/uuid:3926db30-f712-4394-aebc-75976070e91f.
43. Conforti, R., La Rosa, M., ter Hofstede, A.H.M.: Filtering out Infrequent Behavior from Process Event Logs. (2015).