Managing Risk in Process-Aware Information Systems

by
Raffaele Conforti
BEng, MEng

A dissertation submitted for the degree of
IF49 Doctor of Philosophy

Principal Supervisor: A/Prof. Marcello La Rosa
Associate Supervisor: Prof. Arthur H. M. ter Hofstede

Business Process Management Discipline
Information Systems School
Science and Engineering Faculty
Queensland University of Technology (QUT)
GPO Box 2434, Brisbane QLD 4001, Australia

October 8, 2014

Keywords

Risk, risk detection, risk prediction, risk mitigation, workflow, process automation, process model, business process management, process-aware information system, YAWL

Abstract

Business processes in various sectors such as financial, healthcare and resources, are constantly exposed to a wide range of risks. Failures in the management of these risks can result in substantial financial and reputational consequences, potentially threatening an organization's existence. Legislative initiatives in the finance sector have highlighted the pressing need to better manage risks in business processes. As a consequence of these mandates, organizations are now seeking new ways to control risks that may influence business processes and are attempting to incorporate risk management as a distinct view in their operational management. However, whilst conceptually appealing, to date there is little guidance as to how this can best be done.

The objective of this research is to develop a fully-operational approach for the management of risks related to executable business processes in near real-time. This approach consists of four main contributions: i) a language for defining risks on top of a process model at design-time; ii) a technique to detect such risks at run-time, i.e. during process execution; iii) a technique for making risk-informed decisions at run-time, e.g. by suggesting the next process task to perform in order to reduce a given risk; and iv) a technique to automatically mitigate the identified risks at run-time, by performing minimal changes on a running process instance. By incorporating elements of risk in all stages of the lifecycle of executable business processes, this work contributes to create a more effective link between the fields of Business Process Management and Risk Management. The approach was implemented on top of the YAWL system and evaluated though the use of artificial and real-life data.

Contents

List of Figures

List of Tables

Statement of Original Authorship

The work contained in this thesis has not been previously submitted to meet requirements for an award at this or any other higher education institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made. The work contained in this thesis has not been previously submitted to meet requirements for an award at this or any other higher education institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.

8/10/2014

Date

Acknowledgements

I would like to thank my supervisors, Marcello La Rosa and Arthur ter Hofstede for their support during the three years of my PhD studies at Queensland University of Technology. I am thankful to Marcello for his continuous assistance and support, and appreciative of the time spent working together on papers till late at night. I would never have started nor completed this long journey if it were not for him. I am grateful to Arthur who, with his almost unlimited erudition, helped me to understand subtleties in the English language and taught me the importance of an elegant formalization (even though it took me several attempts to get it right).

Special thanks go to Wil van der Aalst from the Eindhoven University of Technology. He was always available to discuss and give good advice on my research. I still remember the first time, just a few months after starting my PhD, when I met him to discuss my research. At that time, Marcello and Arthur were both overseas, so I had to step into his office and introduce myself. While I was rather nervous speaking with him and clueless in terms of on how to address him appropriately, Wil tactfully put me at ease by saying: "do not be so nervous and call me Wil, it is formal enough". During my visits to his group, Wil also gave me the chance to meet Massimiliano de Leoni with whom I had a fruitful collaboration.

I would like to thank all members of the BPM research group. Special thanks goes to Michael Rosemann who, with his expert and inspirational advices, assisted me greatly. I would also like to thank Jan Recker, Moe Thandar Wynn, and Chun Ouyang for their feedback over the years. I would like to thank Michael Adam for his help with the YAWL system during the first few months of my PhD. Moreover, a personal thank you goes to Suriadi, Artem, Jochen, Robert, and Eike for all the time spent together drinking coffee and listening to my complaints.

I would like to thank Giancarlo Fortino, my supervisor during my Master studies, from the University of Calabria who gave me the chance to visit Brisbane and the BPM group, and Peter Hughes for helping me with the identification of

process-related risks.

A personal thank you goes to Hajo Reijers from the Eindhoven University of Technology, with whom I had wonderful conversations during my visit in Eindhoven. Of course, I cannot forget to thank Felix, Daniela, Arya, Joos, Elham, and Dennis who made me feel at home during the months I spent in Eindhoven.

Special thanks go to all my friends here in Brisbane, particularly Ivano, Alfredo, Valentina, Serena, Ali, Thomas, and Stephan. Finally, I would like to thank my family who supported my decision to do my PhD so far away from Italy, and who believed in me all this time.

Chapter 1

Introduction

1.1 Problem Area

In the era of information, the increasing development and adoption of Internet technologies gave companies the opportunity to foster integration and networking. This allowed the emergence of Process-Aware Information Systems, i.e. "software systems that manage and execute operational processes involving people, applications, and/or information sources on the basis of process models" [DAH05].

Process-Aware Information Systems, drastically improved the way companies execute their business processes, for example, by reducing execution time or enforcing standard execution of processes [Gar14, RA05] since process models are executed via process engines. As result of the adoption of PAISs, companies increased the control over the way business processes are executed.

On the one hand, cases like the recent incidents in the finance sector (the €4.9B fraud at Société Générale [Jac10]; the US$2.3B UBS rogue trading scandal [FB13]), in the health sector (the Patel Inquiry [Tho10, Hig]) and in the aviation industry (terrorist attacks [Woo08]) show how business processes are exposed to risks. On the other hand, companies constantly invest money trying to prepare themselves against possible risks for their business activities, since the increased control over a business process, obtained through the adoption of PAISs, does not make it safe.

According to the AS/NZS ISO 31000 standard, a *business process risk* is the chance of something happening that will have a negative impact on the process objectives, and is measured in terms of likelihood and consequence [Sta09]. Legislative initiatives such as the Sarbanes-Oxley Act [10702] and Basel III [Bas11] in the finance sector have highlighted the pressing need to better manage business process risks. As a consequence of these mandates, organizations are now seeking

new ways to *control* process-related risk and are attempting to incorporate it as a distinct view in their operational management [WW11, RBGR06]. However, whilst conceptually appealing, to date there is little guidance as to how this can be done concretely. Currently, the disciplines of business process management and risk management are largely disjoint and operate independently of one another [SWW+14]. In industry they are usually handled by different organizational units, where the disconnectedness of the risk control systems in place provides an environment for potentially high risk exposure.

1.2 Terminology

Several definitions of "risk" [Int09, Sta04, Sta09, SGF01, Swa98] have been proposed in the area of Risk Management. In the context of this research we deal with risks related to business processes, also called *process-related risks*. The definition of process-related risk that we use is inspired by the definition of risk proposed by the AS/NZS ISO 4360 Standard [Sta04] where a risk is "the chance of something happening that will have an impact upon organizational objectives. It is measured in terms of consequences and likelihood".

We base our definition of process-related risk on the definition of *process-related fault*.

> A *process-related fault*, also called a threat, is an unwanted situation that will negatively impact upon the objectives of a business process.

A fault is identified by a condition and a consequence. The condition describes the unwanted situation, while the consequence describes how the fault may impact the process objectives. In particular a consequence is represented using the magnitude of the impact, e.g. the amount of money lost due to the fault. An example of a process-related fault is an overtime fault for a payment process. A Service-Level Agreement (SLA) may establish that the process may not last longer than a Maximum Cycle Time, MCT. An overtime process fault happens if the process execution time exceeds the MCT. The consequence in this case is a fine proportional to the exceeded time.

Figure 1.1 shows the relationship between process-related risk and fault.

> A *process-related risk* is the chance of a process-related fault happening and is measured in terms of consequences and likelihood.

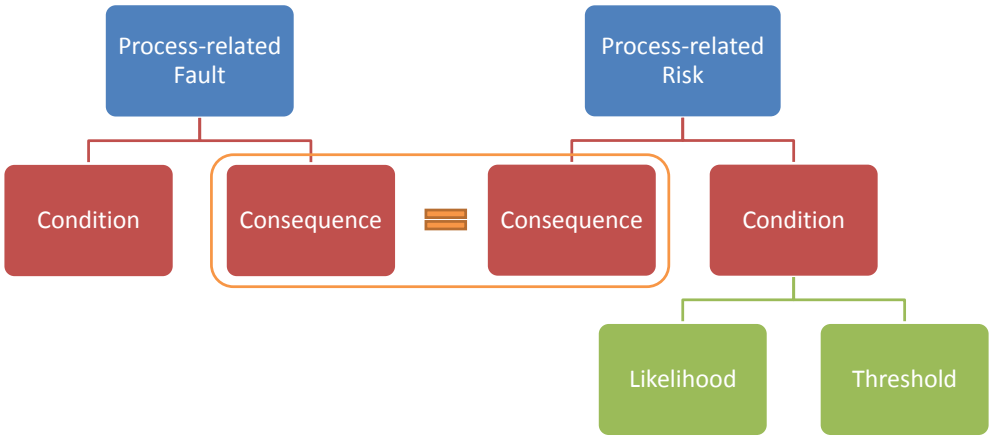


Figure 1.1: Process-related Fault and Risk: Definition and Relationship.

The consequence of a process-related risk is inherited from the fault's consequence since the eventuation of a risk is a fault. The likelihood of a process-related risk is considered in correlation to a threshold. The threshold identifies the limit above which a risk is no longer tolerable for that business process. Finally, though the product of consequence and likelihood for a risk may be relevant for its treatment, it is not strictly required for its detection, thus we did not include this product in the definition of process-related risk. Considering the overtime fault illustrated before, the risk associated with this fault is defined as the probability that the remaining time needed for the completion of the process plus the time used will exceed the MCT.

1.2.1 Business Process Management

Business Process Management (BPM) is a holistic approach whose ultimate goal is improving organizational business processes. BPM has its roots in a number of concepts including Business Process Reengineering, Quality Management (e.g., Total Quality Management, Six Sigma), Operations Management (e.g., Manufacturing resource planning, Lean Management), Business Process Modelling and Process-Aware Information Systems (e.g., Business Process Management Systems and Service-Oriented Architectures).

BPM stresses a process-centered view of the organization. Accordingly, a business process describes a group of activities that need to be executed in a logical and temporal order to pursue a business goal. An example of a business process is a collection of activities which deals with the arrival of an insurance claim all the way to its payment or rejection. With the spread of information technology the execution of business processes becomes possible through the use

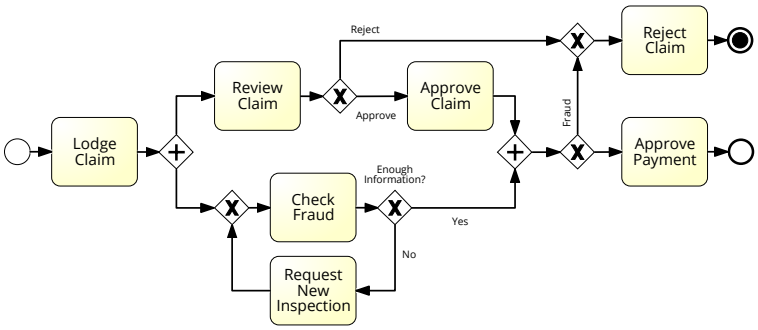


Figure 1.2: Process model of Claim Handling process modelled using the BPMN language

of Business Process Management Systems (BPMSs).

BPMSs are a particular type of PAIS that coordinate the interaction among resources, applications, and external information sources during the execution a business process as a continuous flow of activities [DAH05, HAAR10]. The execution of a business process is guided by a *business process model*. A business process model describes (often via a graphical representation) the relation among activities, resources involved in the execution of these activities, data items processed by these activities, etc. Several description languages have been provided (e.g. BPMN, BPEL, EPC, YAWL), but none of them has been adopted as standard until recently, when in January 2011 the Business Process Model and Notation (BPMN 2.0) language was standardized by the Object Management Group [Obj11], Figure 1.2 shows the model of an claim handling process modelled using the BPMN language.

Though BPMSs, with the automation of business processes, increase organizational productivity and reduce the possibility of committing errors, they cannot assure the correct execution and completion of a business process. During the execution of a business process a *fault* may occur with a resulting loss of profit for the organization. Although the modification and improvement of business processes can protect organizations from faults due to incorrect analysis of their business processes, this is not always the solution. Faults can be caused by exceptions thrown by the BPMS, or by errors and delays introduced by resources. The actual BPMS provide remedial actions for faults caused by the system itself (a feature known as exception handling [AHEA05]). In the case of faults caused by an (un)intentional error made by an external agent (e.g. a resource, a service, etc.) few approaches have been proposed [RMBCO07, MRM12, BKB+14]. They use a planner to generate new process models that will be executed in place of the faulty process instance. Fault prevention is a topic relevant in the Risk Manage-

ment discipline and, in general, faults are prevented with the support of a solid risk analysis [LSS11].

1.2.2 Risk Management

Risk management is a discipline that focuses on the identification, assessment, prevention, and treatment of risks. In the literature, the concept of risk presents several definitions. In general a risk is the chance of something happening that may impact organizational objectives. Several risk standards have been proposed such as those from the Project Management Institute (PMI), the National Institute of Standards and Technology (NIST), and the International Organization for Standardization (ISO) [Sta04, Sta09]. Despite the fact that methods and goals for risk management may vary in relation to context and standard, risk management strategies typically aim to avoid the risk, reduce its negative effect or its likelihood. In general, the prevention of a fault requires: i) the definition of a risk associated with the fault (risk definition), ii) the detection of this risk (risk detection), iii) the prevention of such a risk (risk prevention), and iv) the execution of proper remedial actions (risk mitigation).

Despite Risk Management being a well established discipline [Sta04, SGF01, Hop10, LSS11], it is largely disjoint from BPM and these two disciplines operate independently of one another. In industry they are usually handled by different organizational units. There is no widely recognized, theoretically sound and empirically validated approach for quantifying and managing risks within business processes [SWW+14]. The identification of risks related to business processes can be useful during risk definition but it is inadequate if risk management is not supported during process execution. In fact the simple definition of a process model is not enough if it is not assisted by a mechanism for near real-time detection of process-related risks during the execution of a process. Though risk detection may reduce the effects of a risk when performed in time (e.g. by increasing the level of attention paid during the execution of a process), it becomes much more useful when used in combination with automated risk prevention and risk mitigation strategies.

1.3 Problem Statement

Barring a few exceptions, current research in the area of Risk Management mostly provides generic guidelines for the identification, analysis, evaluation and han-

dling of risks [Sta09] or is specific to a particular area of interest such as financial fraud and investment [AAAZ08, HS65]. Despite the recent efforts undertaken to integrate BPM with Risk Management (see Chapter 2 for a detailed analysis of the literature) there are some lacunae that limit a proper integration of risk management in the context of BPM. Specifically, we have identified the following shortcomings during this research:

Lack of an end-to-end approach for the definition of risks related to business processes, their detection during process execution, their prevention, and their mitigation - Despite the efforts made by academics in recent years, a well defined methodology is missing. Different approaches have been proposed but none of them covers all of the aspects of risk management [SWW+14]. Most of them can be used only for the definition of risks [MH05], or only for their detection [KCK09]. Mitigation when supported only deals with deadline-based or quality of service (QoS) risks [GPL+10, HSD10]. A few approaches propose risk-informed business process design [SGD+08] as a form of risk mitigation. Finally, no approaches were proposed for risk prevention.

Lack of tool support for an end-to-end approach for the definition of risks related to business processes, their detection during process execution, their prevention, and their mitigation - Solutions for process-related risk detection proposed until now present two problems. One class of approaches computes risk probabilities based on current process execution data only [Ora11], i.e. historical execution data is neglected. The other class only provides support for detection of process-related risks through the use general purpose event processing languages [Syb11], or are limited to the detection of abnormal termination [KCK09]. Finally, mitigation, when supported, only deals with deadline-based or QoS risks [GPL+10, HSD10]. Moreover, no approach defines a concrete method usable in the case of process-related risks, e.g. the risk of an underpayment during an order fulfillment. The solutions proposed suggest the use of some predefined remedial actions that will automatically be executed by the BPMS [GPL+10] or the skipping of tasks previously marked as optional [HSD10].

1.3.1 Research Questions

These research addresses the above shortcomings by developing an approach for defining, detecting, preventing, and mitigating process-related risks related to executable business processes. Specially, this research project answers the following research questions:

- How can we formally define process-related risks in order to allow their automatic detection at run-time?
- How can we automatically detect process-related risks during process execution?
- How can we automatically predict risk eventuation?
- How can we automatically provide run-time suggestions to prevent risk eventuation?
- How can we automatically identify remedial actions that can be performed on a running process to mitigate the related risks?

1.3.2 Solution Criteria

The solution resulting from this research will be evaluated using the following criteria:

- Formal: the solution should be supported through a formal definition of the underlying techniques proposed, to provide a clear and unequivocal interpretation of the solution.
- Executable: the solution should be able to be performed during the execution of a business process.
- Acceptable performance: detection and mitigation of process-related risks as well as risk-informed prediction should be executed in an amount of time that is acceptable for near real-time application. In other words, these operations should provide a timely response whenever they are utilized.
- Comprehensive: the solution should be able to address all identified issues for the management of process-related risks, i.e. process risk definition, detection, prevention and mitigation.
- Language and Tool Independent: the solution should be applicable without restrictions to a specific process modeling language or BPMS.

1.3.3 Research Benefits and Innovation

The importance of this research is emphasized by the need of many organizations to manage business processes, as shown by Gartner's 2013 survey [Gar14].

Gartner asked 2,053 CIOs in 41 countries, representing more than $230 billion in CIO IT budgets, what were the Top 10 CIO Business and Technology Priorities in 2013. The results show that six out of ten top priorities are related to the management of business processes and "improving business processes" is the 10th business priority. This management must, however, be governed by appropriate risk management and mitigation strategies. Organizations will benefit from this research, since they will be able to reduce economic and/or reputational losses caused by risks that happened during the automated execution of business processes and that negatively impact their business. In particular, the results of this research will allow companies to have a better understanding of their process-related risks. Companies will also be supported by techniques and tools that will allow a timely and automatic resolution of process-related risks, that will ultimately result in an improvement of the reliability of automated business processes. Moreover, this research is innovative because it is the first time that:

- A functional (i.e. supported by tools) link between risk management and business process management is realized.
- A technique is devised for providing operational support aimed at the prevention of risk during the execution of business processes.
- A technique is devised for automating the mitigation of process-related risks during the execution of a business process.

1.4 Research Approach

The purpose of this research is to devise and operationalize a holistic approach for the near real-time management of process-related risks. In this context *near real-time* refers to the capability of the approach to operate during the execution of a business process within minutes from the occurrence of a trigger [Wer78]. Specifically, this approach provides support for the definition, and near real-time detection, prevention, and mitigation of risks related to running business processes.

The research approach of this project is design science [HMPR04]. Design science is based on two pillars: relevance and rigor. The term *relevance* is used to address research that introduces new knowledge on a topic, proposing a novel and better approach to a problem or a solution to an unsolved problem. The

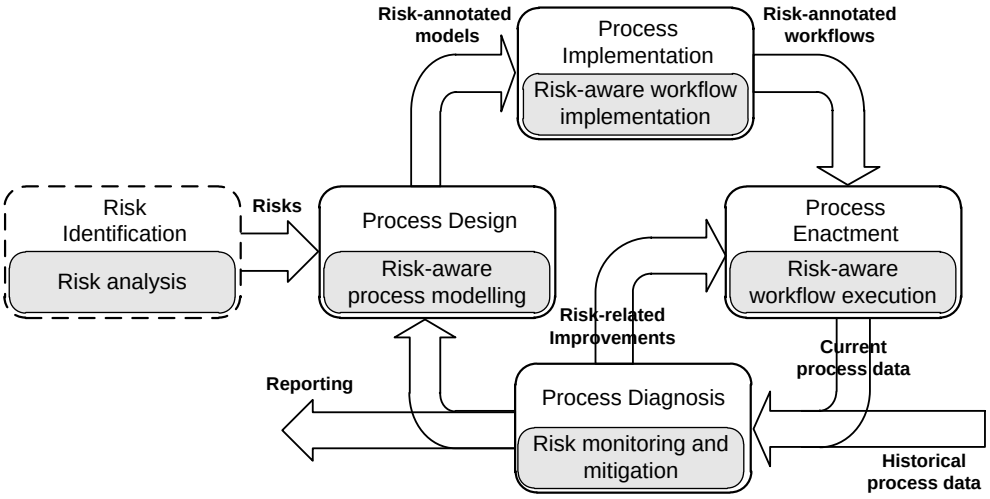


Figure 1.3: Risk-aware Business Process Management lifecycle.

term *rigor* is used to identify a scientific approach usually achieved by applying established foundations and methodologies. This research will be relevant because both practitioners and academics have recognized the importance of integrating the concept of risk (and its management) into the BPM discipline, and academics are trying to fill this gap [NCMR06, MH05, TJQ08]. The rigor of this research is guaranteed by the use of formal methods, by the implementation of the envisaged concepts in prototype tools, by the use of (but not limited to) well established languages such as YAWL and BPMN, and the empirical evaluation of each technique.

This research embeds elements of risk into all four phases of the lifecycle of executable process models in the context of PAISs [DAH05], as shown in Figure 1.3. Input to this "risk-aware" BPM lifecycle is a *Risk Analysis* phase, where risk analysis is carried out to identify risks in the process model to be designed. Traditional risk analysis methods such as Fault Tree Analysis [Com90], Root Cause Analysis [Joh73] or CORAS [LSS11], can be employed in this phase. The output of this phase is a set of risks, each expressed as a risk condition. It is important to note that not all risks identified during risk analysis can be avoided by preemptively modifying the business process in which they may eventuate, and this is why near real-time risk management is required. For example, the risk associated with an overtime fault cannot be prevented through a preemptive modification of the business process since a delay may occur independently of the process. Next, in the *Process Design* phase, these high-level risks are mapped down to process model-specific aspects. The result of this second phase is a risk-annotated process model. In *Process Implementation*, these risks are linked

Figure 1.4: Research Phases.

to workflow-specific aspects, such as values of variables, and resource allocation statuses. The risk-annotated process model resulting from the Process Implementation phase is then executed by a risk-aware process engine during *Process Enactment*. In this phase "decision support" for risk prevention is provided by suggesting which decision to take in order to keep the level of risk under control, during the execution of a process instance. Finally, historical data stored in process logs, and current execution data coming from process enactment, is filtered, aggregated and analyzed in the *Process Diagnosis* phase, in order to evaluate the possibility that a risk may happen. When a risk is detected, the system will automatically identify an adequate remedial action to mitigate the likelihood of a fault to occur.

This research covers four aspects of the management of process-related risks. These four aspects (see Figure 1.4) and the phases where they are performed are: risk definition during process design, risk detection during process diagnosis, risk prediction during process enactment, and risk mitigation during process diagnosis. Firstly, a formal method to specify risks is defined. On top of it, an interpreted language (i.e. a language that a machine can understand and convert in binary code at run-time) for the definition of risks in a business process (i.e. risk definition) is defined. Based on this language an architecture for the detection of risks is devised and implemented (i.e. risk detection). The architecture is based on sensors. Such sensors monitor the execution of business processes and detect risks (defined using the interpreted language). The detection of risks is carried out through the analysis of information related to the current executions (activities' variables, activities' status, resources allocation status, etc.) and information related to previous executions (i.e. historical data that has been logged by the process engine).

The following step is the definition of a method for risk prediction and risk-informed business process execution. The idea is to guide at the user during the execution of a business process, suggesting which step to execute in order to reduce the likelihood of risks occurring. We present a method that predicts the future status of a process instance, via decision trees, trained using historical

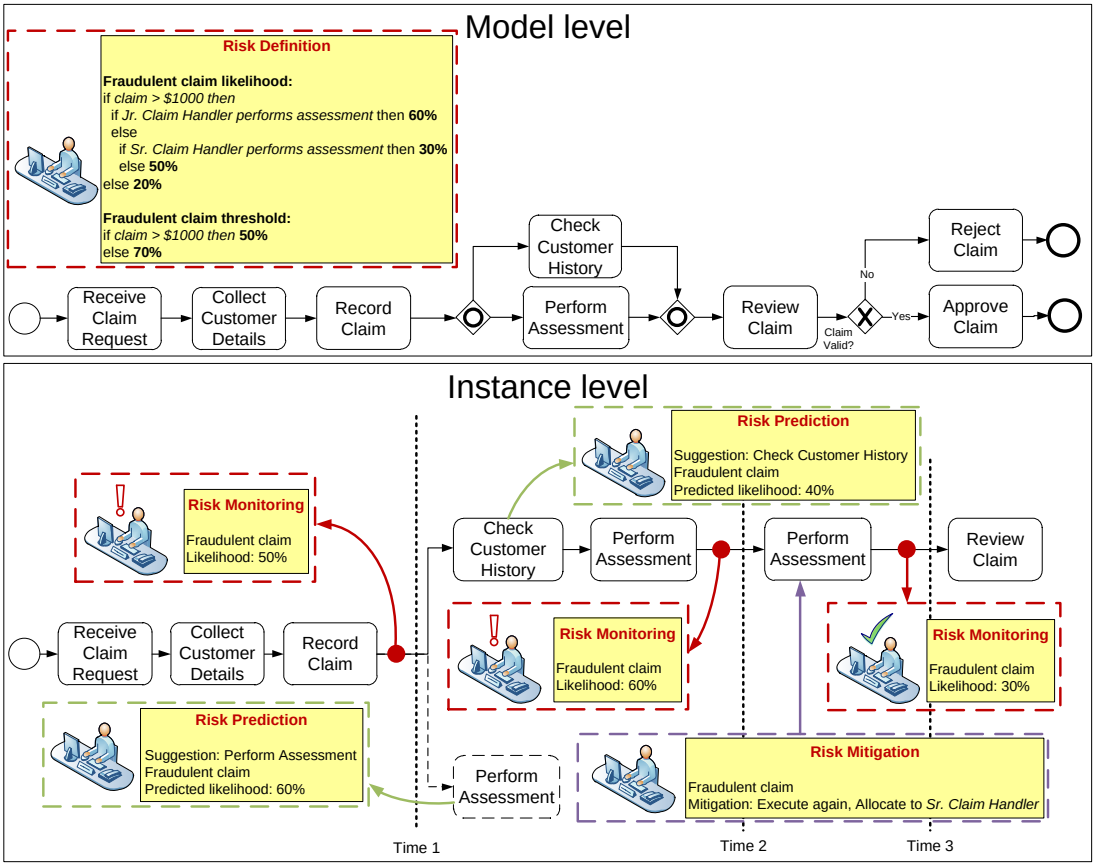


Figure 1.5: Process execution with risk support.

data (i.e. completed process instances), and with it the values of the risk conditions associated with the process instance. This information is then used to solve an optimization problem, in particular a mixed integer linear programming problem, where the goal is minimizing risk eventuation. The solution discovered is then used to provide suggestions to process participants about what activity they should perform. Finally, a prototype of this method is developed. After integrating this prototype with the risk detection architecture, evaluations are carried out to measure the feasibility of the approach, verifying in the end if the requirements of the project are met.

Finally, this research presents a method for risk mitigation. Mitigations are discovered using the simulated annealing algorithm. In particular, we first identify a set of controlled changes that can be applied to a process instance. We then defiy functions that are used for the identification of the perturbations (i.e. controlled changes) that can guarantee a (sub-)optimal mitigation. Finally, based on the results of this analysis a prototype for automatic risk mitigation is defined and realized. After integrating this prototype with the risk detection architecture, evaluations are carried out to measure the feasibility of the approach, verifying

in the end if the requirements of the project are met.

Figure 1.5 shows (using the BPMN language) how our approach supports risk management during the execution of a business process. Let's assume we have an claim handling process that is exposed to the risk of fraudulent claims. We define the risk as equal to 20% for claims below $1000, while for claims over $1000 the risk is 60% if the assessment is performed by a junior employee, 30% if the assessment is performed by a senior employee, and 50% if the assessment is not performed. Moreover, the risk has a tolerance threshold defined in such a way that it is equal to 70% for claims below $1000 and 50% otherwise. After defining the risk, we can start the execution of the process. After receiving a new claim for $5000 and collecting the details of the customer, we proceed recording this information in the database. Here, our risk monitoring technique detects that the claim may be fraudulent with a likelihood of 50%, and proceeds with notifying the Process Administrator and the Process Participants. At this point in time we have to check the history of the customer (if the customer has one) and perform the assessment. In this case, our approach will suggest to Process Participants to perform the check (lower predicted likelihood) before the assessment, because an employee aware of previous claims could then easily identify non-claimable damages during the assessment. Next, the assessment is performed by a Junior Claims Handler, in which case the monitoring technique detects that the risk has now increased to a likelihood of 60%. Here the Process Administrator can request our approach to find a possible mitigation, that will reassign the assessment of the claim to a Senior Claims Handler, given the high likelihood of this claim being fraudulent.

Finally, the techniques presented in this thesis are realized on top of the YAWL system[1] [HAAR10]. We decided to extend the YAWL system for the following reasons. First, this system is based on a service-oriented architecture, which facilitates the seamless addition of new services. Second, the system is open-source, which facilitates its distribution among academics and practitioners (the system has been downloaded over 100,000 times since its first inception in the open-source community). Finally, the underlying YAWL language is very expressive as it provides support for the workflow patterns [HAAR10].

[1] Available at `http://www.yawlfoundation.org`

1.5 Research Scope

In the context of this research we do not propose techniques for the assessment of process-related risks. Instead, we use established risk analysis techniques present in the literature such as Fault Tree Analysis. Also our approach does not identify unexpected risks but only risks identified during the risk assessment phase. For this reason, we are not be able to provide any mitigation in case of unexpected risks.

In Figure 1.3, a connection between the Process Diagnosis phase and the Process Design phase is shown. This connection shows that the Process Diagnosis phase may, in principle, influence the redesign of a process, e.g. by removing or mitigating certain process risks by design. How this can be achieved falls outside the scope of this research. Similarly, post-execution analysis (during the Process Diagnosis phases) are not addressed in this work since the scope of this research is providing an approach for managing risks during process execution.

The context in which a risk may manifest itself is also interesting and worthy of further investigation. However, in this research we focus on process-related risks that can be identified within the boundaries of a business process. In particular, we only consider process-related risks which depend on information available during process execution, e.g. task input and output data, allocated resources, time performance. This implies that process-related risks depending on information outside the process boundaries, i.e. the process context (e.g. market fluctuations or weather forecast) cannot be detected. For this reason organizational risks in general are not addressed, such as those related to partners going bankrupt, or the price of fuel going up. Moreover, since we require process execution information we only consider "executable" business processes. These processes should either be executed by a BPMS on the basis of a process model or be supported by an information system that produces event logs [Aal11], i.e. logs of process-related information which we can use to reconstruct the process instances being executed by aggregating events, such that each instance can be unequivocally identified.

Finally, in our approach a risk is detected when its likelihood exceeds a user-defined threshold. The product of likelihood and consequence is not be considered for the detection of a risk since we assume that a risk's threshold (e.g. 70%) itself is an indication of the degree of risk an organization is willing to accept. This assumption does not, however, prevent us from considering this product during the mitigation of a risk, where mitigating risks with high likelihood and consequence is preferable to mitigating risks with high likelihood but low consequence.

1.6 Publications

This research led to the following publications:

- R. Conforti, G. Fortino, M. La Rosa, A.H.M. ter Hofstede. History-aware, real-time risk detection in business processes. In R. Meersman, T. Dillon, P. Herrero, A.Kumar, M. Reichert, L. Qing, B.C. Ooi, E. Damiani, D.C. Schmidt, J. White, M. Hauswirth, P. Hitzler, M.K. Mohania (Eds.) *Proceedings of the 19th International Conference on Cooperative Information Systems (CoopIS'11)*, Hersonissos, Crete, Greece. Lecture Notes in Computer Science Vol. 7044, pp 100-118. Springer, 2011.

- R. Conforti, A.H.M. ter Hofstede, M. La Rosa, M. Adams. Automated Risk Mitigation in Business Processes. In R. Meersman, H. Panetto, T. Dillon, S. Rinderle-Ma, P. Dadam, X. Zhou, S. Pearson, A. Ferscha, S. Bergamaschi, I.F. Cruz (Eds.) *Proceedings of the 20th International Conference on Cooperative Information Systems (CoopIS'12)*, Rome, Italy. Lecture Notes in Computer Science Vol. 7565, pp 212-231. Springer, 2012.

- R. Conforti, M. La Rosa, G. Fortino, A.H.M. ter Hofstede, J. Recker. Real-Time Risk Monitoring in Business Processes: A Sensor-based Approach. *Journal of Systems and Software*, 86 (11), pp 2939-2965, 2013.

- R. Conforti, M. de Leoni, M. La Rosa, W.M.P. van der Aalst. Supporting Risk-Informed Decisions during Business Process Execution. In C. Salinesi, M.C. Norrie, O. Pastor (Eds.) *Proceedings of the 25th International Conference of on Advanced Information Systems Engineering (CAiSE'13)*, Valencia, Spain. Lecture Notes in Computer Science Vol. 7908, pp 116-132. Springer, 2013.

- R. Conforti, M. La Rosa, G. Fortino. Process Monitoring Using Sensors in YAWL. In T. Freytag, A. Hense, A.H.M. ter Hofstede, J. Mendling (Eds.) *Proceedings of the First YAWL Symposium*, Sankt Augustin, Germany. CEUR Workshop Proceedings Vol. 982, pp 49-55, CEUR, 2013.

- R. Conforti, M. La Rosa, A.H.M. ter Hofstede, G. Fortino, M. de Leoni, W.M.P. van der Aalst, M. Adams. A Software Framework for Risk-Aware Business Process Management. In R. Deneckère, H.A. Proper (Eds.) *Proceedings of the CAiSE'13 Forum at the 25th International Conference on Advanced Information Systems Engineering (CAiSE'13)*, Valencia, Spain. CEUR Workshop Proceedings Vol. 998, pp 130-137, CEUR, 2013.

- S. Suriadi, B. Weiß, A. Winkelmann, A.H.M. ter Hofstede, M. Wynn, C. Ouyang, M. Adams, R. Conforti, C. Fidge, M. La Rosa, and A. Pika. Current research in risk-aware business process management - overview, comparison, and gap analysis. *Communications of the Association for Information Systems*, 34 (1), Article 52, 2014.

- R. Conforti, M. de Leoni, M. La Rosa, W.M.P. van der Aalst, A.H.M. ter Hofstede. A Recommendation System for Predicting Risks across Multiple Business Process Instances. BPM Center Report BPM-14-04, BPMcenter.org, 2014.

1.7 Outline

This thesis is organized as follows. Chapter 2 reviews existing literature on risk-aware business process management, and positions the research presented in this thesis. Chapter 3 provides the formal foundations for various notions that will be utilized throughout this thesis, such as the definition of a process model, specifically a YAWL specification, the conceptualization of an event log resulting from the execution of a business process, and an introduction to optimization techniques such as linear programming and simulated annealing. Chapter 4 introduces a technique and corresponding tool implementation for risk definition and risk monitoring based on sensors. Chapter 5 presents a technique and corresponding tool implementation for predicting risk eventuation. Chapter 6 resents a technique and corresponding tool implementation for risk mitigation based on the use of simulated annealing, as optimization technique. Finally, Chapter 7 concludes this thesis summarizing the work presented and discussing possible avenues for future work.

Chapter 2

Literature Review

This chapter reviews existing literature on risk-aware business process management, and positions the research presented in this thesis. In particular, Section 2.1 describes the state of the art of business process management. Section 2.2 presents risk management and some methodologies and methods proposed, with a focus on those proposed in the context of business process management. Section 2.3 briefly introduces business rules. Section 2.4 describes business process monitoring and in particular two possible methods for its realization (i.e. Business Activity Monitoring and Complex Events Processing). Section 2.5 presents business process adaptation and exception handling. Section 2.6 discusses on business process improvement. Finally, Section 2.7 outlines business intelligence and its application to business processes with particular a emphasis on business process mining. This chapter elaborates on and extends work published elsewhere [SWW+14].

2.1 Business Process Management

Improving the performance of business processes has always been one of the main foci for organizations. Moreover the possibility of improving business processes in a short amount of time is desirable not only from the point of view of resource consumption (regarding the time spent to modify the process) but also because a quick adaptation of the process can bring a quick improvement to the business. This need brought about the birth of business process management (BPM) defined by Dumas et al. [DLMR13] as: "the art and science of overseeing how work is performed in an organization to ensure consistent outcomes and to take advantage of improvement opportunities".

Companies that want to engage in a BPM initiative need to go through several phases (see Figure 2.1): i) process identification, where relevant processes

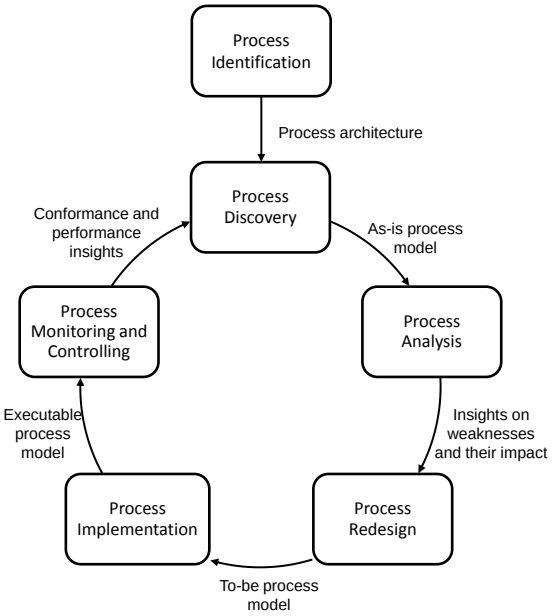


Figure 2.1: The BPM lifecycle [DLMR13].

are identified and related to each other; ii) process discovery, in which the current state of each of these processes is documented (as-is processes); iii) process analysis, where issues related with as-is processes are identified; iv) process redesign, in which such issues are addressed through changes in these processes; v) process implementation, where to-be processes are created to allow their automation; finally vi) process monitoring and controlling in which data related to such processes is collected and analyzed in order to identify bottlenecks, errors, and deviations from the intended behavior.

A business process has to go through an additional phase in order to be executed. Van der Aalst et al. [AHW03] identified four phases in the lifecycle of an automated business process (see Figure 2.2): i) process design, which corresponds to the process discovery phase of the BPM lifecycle; ii) process implementation; iii) process enactment, in which the business process model is executed; and iv) diagnosis, which corresponds to the process monitoring and controlling phase. In this thesis when we refer to the BPM lifecycle we refer to the lifecycle for automated processes.

Ter Hofstede et al. [HAAR10] discuss the role of models in BPM. In particular they identify three main purposes for a model:

- Providing *insight*: Process models can be used to provide a clear overview of aspects related to the process to different stakeholders involved in the process. For example, process models can be used to discuss requirements,

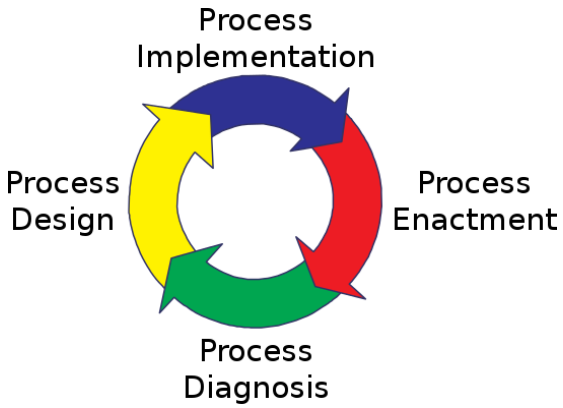


Figure 2.2: The lifecycle of automated processes inspired by [AHW03].

to support design decisions, validate assumptions, etc.

- *Analysis*: Process models can be analyzed, depending on the type of model, to check the performance of the system or to detect errors and inconsistencies in the process.

- Process *enactment*: Process models can be used to execute business processes. Based on the languages used for their definition the models can be directly executed on BPM systems or be used for automatic creation of a machine-readable format.

2.1.1 Business Process Modeling Languages

Different business process modeling languages have been defined for the creation of business process models. Weske [Wes07] discussed the constructs used in several modeling languages. Different modeling languages support different levels of abstraction. Languages created with the purpose to facilitate requirement analysis and communication focus on capturing conceptual business processes and do not necessary include non-functional information. Executable languages must encode business processes using a format that can be interpreted by a machine, i.e. a process engine. In general, executable languages use Extensible Markup Language (XML) or are XML-like. The main languages used for process modeling are: Business Process Model and Notation (BPMN) [Obj08a], extended Event-driven Process Chains (eEPCs) [DB07] and Unified Modelling Language (UML) activity diagrams [Obj09a, Obj09b]. Languages that support process enactment are Yet Another Workflow Language (YAWL) [HAAR10] and the Business Process Execution Language (BPEL) 2.0 [AAB+05].

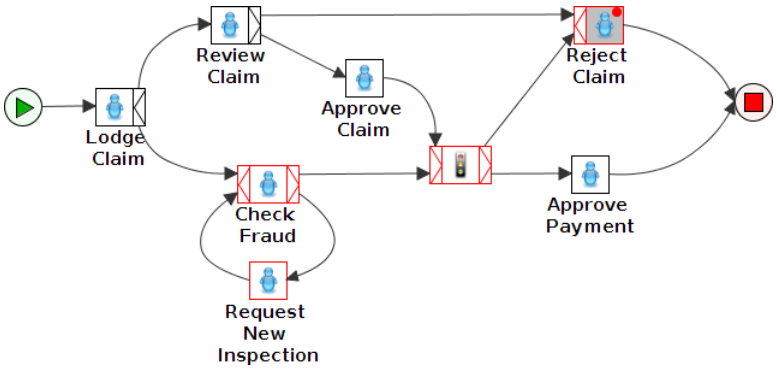


Figure 2.3: YAWL model of the Claim Handling process shown in Figure 1.2.

Using these languages it is possible to model activities, events, gateways, data objects and participating resources to various degrees of abstraction. The activity construct models an operation (task) inside a business process. Depending on the language used, it is possible to define different types of tasks: atomic task, composite task (used for subprocesses), and resource task. The event construct models the occurrence of situations which comes from outside and which can activate a process or an activity. In general the events supported by process modeling languages are: trigger events, message events, and error events. The gateway constructs are used to model the flow of the process, beyond a simple sequence of activities. Typical gateways constructs are: AND-split, AND-join, OR-split, OR-join, XOR-split, XOR-join which allow various forms of branching, merging and synchronization. Data objects manage information used for the execution of an activity. In general, data objects can be defined at the activity level or at the (sub)process level. Often an activity is to be performed by a resource (also named actor, performer, originator or participant). In such a case, the model may also specify the resource allocation strategy (e.g. offered by the system, allocated by the system, started by the system, etc.). The allocation strategy can refer to a specific resource or to a role or capability.

Finally, Figure 2.3 again presents a model of the claim handling process presented in the Introduction, this time using the YAWL language. As previously specified, in a process model we may have several activities (or tasks) that need to be executed. Example of tasks are: review claim, approve claim, check fraud, etc. Processes under execution are referred to as process instances or cases and it is possible to have several instances of the same process. During the execution of a process instance every time a task needs to be executed a work item is created. A work item is an instance of a task, and in the context of a process instance it is possible to have several work items for the same task.

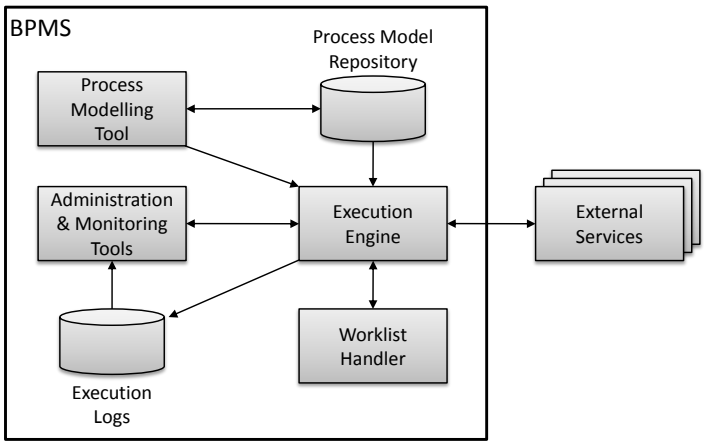


Figure 2.4: Architecture of a BPMS [DLMR13].

2.1.2 Workflow Reference Model

Business process models are executed using Business Process Management Systems. Figure 2.4 shows the main components that constitute the architecture of a BPMS. The core of the architecture is the execution engine. It provides a run-time environment where instances of processes can be executed. An execution engine takes care of creating process instances, distributing work to process participants, and retrieving and storing data required for the execution. The execution engine interacts with a process modelling tool, which can store and retrieve process models from a process model repository. The execution engine does not directly interact with process participants but instead uses a worklist handler. Finally, it can also interact with external services, for example in case of automated activities.

In Figure 2.5 the meta-model for process definition as proposed by the Workflow Management Coalition (WfMC) is shown. This model identifies a set of object types for the representation and interchange of process definitions. In this model the WfMC identifies six main object types that can be extended and integrated with other types, based on the necessity of the vendors. These six types are: i) *Workflow Type Definition* containing workflow process name, version of the process, starting condition, terminal condition, and control data; ii) *Activity* containing activity name, activity type, pre- and post- conditions, and scheduling constraints; iii) *Transition Conditions* containing flow or execution conditions; iv) *Workflow Relevant Data* containing data name and path, and data types; v) *Role* containing name and organizational entity; vi) *Invoked Application* containing generic type or name, execution parameters, and location or

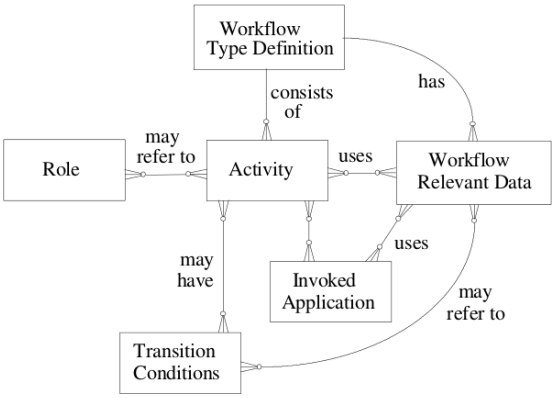


Figure 2.5: Basic Process Definition Meta-model [Hol95].

access path.

2.2 Risk Management

The term "risk" is often used in many contexts and domains, and different definitions of risk can be found in the literature. The AS/NZS ISO 4360 Standard [Sta04] defines a risk as "the chance of something happening that will have an impact upon objectives. It is measured in terms of consequences and likelihood". It is common to see it linked to different kinds of risks (e.g. economic risk, environmental risk, security risk, etc.). The need to manage these different risks led to the birth of different notions like "risk analysis", "risk evaluation", "risk treatment". All of these concepts can be grouped under Risk Management [Sta09, Int09]. Risk management is defined as "the systematic application of management policies, procedures and practices to the tasks of establishing the context, identifying, analyzing, assessing, treating, monitoring and communicating risk" [Sta04]. Figure 2.6 shows an overview of the risk management process according to the Australia and New Zealand Standards [Sta04]. The main elements constituting this process are described here:

- *Establish the context*: identifies the contexts (i.e. strategic context, organizational context, risk management context) in which the risk management process is adopted, establishes the criteria for the risk evaluation and defines the structure of the analysis.

- *Identify risks*: identifies possible threats that can constitute risks, also for risks that are not under the control of the organization.

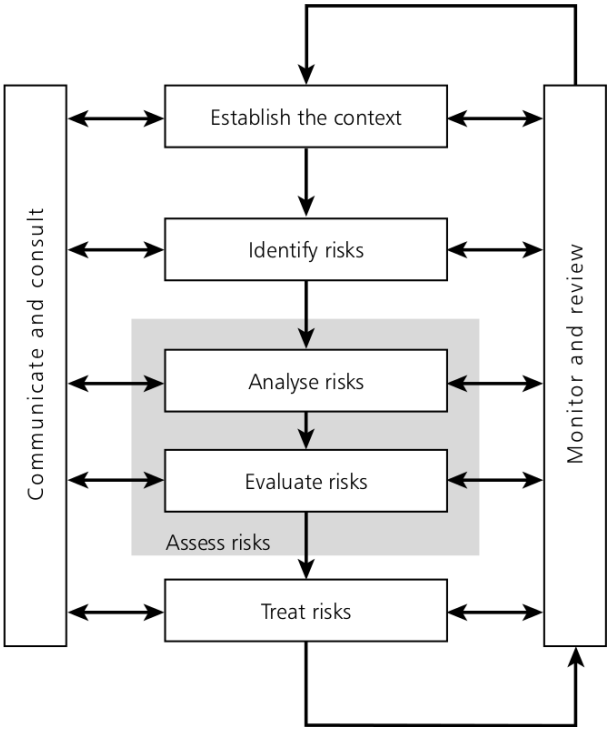


Figure 2.6: Risk Management overview [Sta04].

- *Analyze risks*: determines consequences and likelihood of risks identified.
- *Evaluate risks*: ranks risks using pre-established criteria according to their severity in order to decide whether to treat a risk or not.
- *Treat risks*: apply selected treatments to high-priority risks.
- *Monitor and review*: monitors and reviews the performance of the system and the changes that might affect it.
- *Communicate and consult*: communicates and consults with internal and external stakeholders about the process as a whole.

The first five elements of the risk management process are identified as the *risk analysis process*. Under the heading "risk analysis" one can find different approaches [LSS11]. Usually it is possible to group these approaches into two main categories: i) Offensive approaches concerned with balancing potential gains against eventual risks associated with a certain choice; ii) Defensive approaches concerned with preserving actual conditions against unexpected or unwanted external interactions.

Lund et al. [LSS11] introduce another classification for risk analysis approaches. They subdivide these approaches into two groups: risk analysis methods and risk

analysis techniques. A risk analysis method provides advice on the performance of all stages of the risk analysis process, while a risk analysis technique only provides advice on some of these.

2.2.1 Risk Analysis Methods and Techniques

As mentioned before, risk analysis methods provide support during all the five steps of the risk analysis process (i.e. establish the context, identify risks, estimate risks, evaluate risks, treat risks) while risk analysis techniques only provide support for some phases of the risk analysis process. For example, the following approaches can be classified as risk analysis methods: CORAS [LSS11], Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) [AD01], and CCTA Risk Analysis and Management Method (CRAMM) [BD92]. Risk analysis techniques are: Hazard and Operability (HazOp) [Int01], Failure Mode Effect Analysis/Failure Mode Effect and Criticality Analysis (FMEA/FMECA) [US 49], Fault Tree Analysis (FTA) [Com90], Event Tree Analysis (ETA) [Com95], and Cause-Consequence Analysis (CCA) [Nie71].

CORAS is an asset-driven approach which focuses on defensive risk analysis providing a UML-like graphical language for threat and risk modeling. An asset for CORAS is something valuable that a "party" (e.g. organization, company, person, etc.) wants to protect. In CORAS, risk analysis is conducted in eight steps: i) *Preparations for the analysis* during which the target and the scope of the analysis are identified; ii) *Customer presentation of the target* is the introductory meeting with the customer (i.e. who will benefit from the risk analysis), to achieve a common initial understanding of the target; iii) *Refining the target description using asset diagrams* aims to ensure a common understanding of the target including its focus, scope and main assets; iv) *Approval of the target description* aims to ensure that the background documentation for the rest of the analysis is correct and complete; v) *Risk identification using threat diagrams* aims to conduct a systematic identification of threats, unwanted incidents and vulnerabilities [LSS11] linked to assets using structured brainstorming; vi) *Risk estimation using threat diagrams* aims to determine the risk level of the identified risks; this phase is carried out using brainstorming and produces an estimation of likelihoods and consequences of unwanted incidents; vii) *Risk evaluation using risk diagrams* aims to decide which of the identified risks are acceptable and which need treatment; viii) *Risk treatment using treatment diagrams* aims to identify and analyze treatments for unwanted situations; this phase should produce a

treatment that reduces likelihood and/or consequences of an unwanted situation.

OCTAVE is an approach for risk-based information security strategic assessment and planning. The OCTAVE approach focuses on two aspects: operational risk and information security practices. This approach is defined through a set of criteria. These criteria are principles, attributes, and outputs. Principles are the concepts driving the evaluation process. Attributes and outputs identify the requirements of the evaluation. Attributes are derived from the OCTAVE principles and define what is necessary to make the evaluation a success. Outputs define the outcomes that must be achieved during the evaluation. OCTAVE is composed of three phases. The first phase (*Build Asset-Based Threat Profiles*) involves the evaluation of the company's security strategy and identifies critical assets and threat profiles. The second phase (*Identify Infrastructure Vulnerabilities*) involves the evaluation of the information infrastructure to determine key components and vulnerabilities. Finally, the third phase (*Develop Security Strategy and Plans*) uses the information collected during previous phases to assess the risk of data compromise and the risk associated with the company's business activities. All these phases are discussed during workshops using structured brainstorming and the results are documented using tables and threat trees.

CRAMM is based on the UK Government's Risk Analysis and Management Method. In this approach risk analysis covers the aspects related to the identification and assessment of security risks, while the identification of adequate treatments of these risks is solved by risk management. The CRAMM method has three phases, the first two phases focus on risk analysis while the last phase focuses on risk management. The first phase (*The establishment of the objectives*) aims to the identification and the valuation of the assets that are part of the system. The second phase (*The assessment of the risks*) calculates measures of risks using threats and vulnerabilities identified in the previous phase. Finally, the third phase (*Identification and selection of countermeasures*) identifies countermeasures that are commensurate with the measures of risks calculated in the second phase.

These three methodologies (i.e. CORAS, OCTAVE, and CRAMM) can be indiscriminately used in our context for the identification of risks, as this research does not intend to support risk identification but requires it for risk detection, prevention, and mitigation. Though using different names and different phases, these three methodologies focus on three main elements: i) identification of risks

using brainstorming sessions; ii) evaluation of criticality of the risks identified; iii) identification of treatments for the risks identified. While the first two elements require support from one of these methodologies, for the identification of treatments we use the methods that will be developed as part of this research.

HazOp is a technique for risk identification that requires gathering a number of participants that through brainstorming identify hazards related to a target system. The brainstorming sessions are structured around system documentation and are driven by the HazOp leader through the use of a set of guide words used to formulate questions to the participants about components of the system. The results are presented in tables that document the hazards identified.

FMEA/FMECA is a bottom-up technique that focuses on identifying possible failure modes and determining their consequences. The identification is carried out through structured brainstorming. Once all the failure modes have been identified, the level of risk and the probability of the failure modes are quantified. The results are documented using FMEA/FMECA tables that highlight failure modes, effects and criticality for each component.

FTA is a top-down risk analysis technique based on fault-trees. A fault-tree is a graphic notation to model a pathway that, connecting events and conditions with standard logic symbols (i.e. AND, OR), leads to an undesirable event. Once the fault-tree is defined, the probability of an unwanted event can be calculated using a bottom-up approach by propagating probabilities of leaf nodes through the gates (i.e. standard logical symbols).

ETA is a bottom-up risk analysis technique that uses a tree notation to represent the outcome and the probability of an event. The tree is created starting from an unwanted event, which is connected with a success or failure conclusion through a set of branches. A new branch is created at each decision/action that must be taken/executed. This technique can be used for qualitative analysis (showing outcomes for an event) and for quantitative analysis (estimating the likelihood of each outcome).

CCA is a graph-based technique that combines features of both fault and event trees. The graph is constructed starting from an unwanted event. From this event the diagram is developed finding causes (as a fault tree) and consequences (as an event tree) of the event.

Among these five methods for the identification of causes/effects of risks, the first two approaches are not relevant for our research because they are based on brainstorming sessions while we require our solution to be executable. The last three methods, on the other hand, are quite relevant, in particular the FTA, since it provides a logical representation of risks based on conjunctions of simple elements.

2.2.2 Process-Related Risk Management

Previous process-based research recognizes the importance of explicitly linking elements of risk to business process models. In this section, based on Suriadi et al. [SWW+14], these approach are discussed and evaluated against a set of criteria. These criteria measure if a given approach covers the BPM lifecycle providing risk-aware support in each of its phases and what type of support they provide, i.e. risk identification, risk monitoring, risk prevention, and risk mitigation.

The list of approaches taken in consideration is shown in Table 2.1. We categorized these approaches into three groups. In particular we categorized them as approaches that mainly work at i) design-time, ii) run-time, and iii) post-execution. The approach code column shown in Table 2.1 indicates the group to which each of these approaches belongs: those approaches with an "approach code" starting with the letter "D" belong to the first group, those with "RT" belong to the second group, and those with "PE" belong to the third group.

DI01: The Risk-Oriented Process Evaluation (ROPE) methodology [JGTQ08, TJQ08, TJGK10, TJGQ08, TJG+11, JTQ07, JTGK10a, JNT09, EFKW07, JT09, JTGK10b, GEF+08] proposed by Jakoubi et al. is a methodology that captures the notion of "risk" in business management with the purpose of making business processes safer. The ROPE methodology is composed of five iterative steps. The first step is the *Strategic Decision Process*. During this phase all the business processes that are relevant for the company are identified and prioritized. The second step is the *Re-Engineering Process*. This phase consists of five sub processes (i.e. criteria selection stage, acquisition stage, analysis phase, design stage, and evaluation stage) which use the criteria defined during the first step to develop a target model (i.e. a model that can be used for simulation) of the business model. The third step is the *Resource Allocation Process*, where resources required for the execution of the business process are identified, assigned, and coordinated. The fourth step is the *Workflow Management Process*. In this phase

Authors	Code	Reference(s)
Jakoubi et al., Tjoa et al.	DI01	[TJQ08, TJGK10, TJGQ08, TJG+11, JTQ07, JTGK10a, JTGK10b, JNT09, EFKW07, JT09, JGTQ08, GEF+08]
Sienou et al.	DI02	[SLPK10, SLPK08, SLKP07, SLPK09, SLP08, SKP06, SKLP08, KSLP07, Sie09]
Cope et al.	DI03	[CKE+10, CKE09, CDE+10]
Weiß and Winkelmann	DI04	[WW11]
Asnar and Giorgini	DI05	[AG08]
Mock and Corvo	DI06	[MC05]
Rosemann and zur Muehlen	DI07	[RM05]
Rotaru et al.	DI08	[RWCN08, NCMR06, RWC+11]
Betz et al.	DI09	[BHO11]
Herrmann and Herrmann	DI10	[HH06]
Strecker et al.	DI11	[SHF10]
Karagiannis et al.	DI12	[KMS07]
Taylor et al.	DI13	[TGM08]
Panayiotou et al.	DI14	[POAG10]
Lambert et al.	DI15	[LJJ06]
Bai et al.	DI16	[BBK06, BPK07, BPK06]
Bhuiyan et al.	DN01	[IBKG09, BIK+07]
Fenz et al.	DN02	[FN09, FEN09, FE09, Fen10]
Kaegi et al.	DN03	[KMZN06]
Bergholtz et al.	DN04	[ABE+05, BGJ+05, SGD05]
Jallow et al.	DN05	[JMV+07]
Singh et al.	DN06	[SGD+08]
Kang et al.	RT01	[KCK09]
Jans et al.	PE01	[JWLV11, JLV01, JDV11]
Wickboldt et al.	PE02	[WBL+11]
Pika et al.	PE03	[PAF+13a, PAF+13b]
Suriadi et al.	PE04	[SOAH13]

Table 2.1: Risk-Aware BPM Approaches - Authors, Code, and References.

the business process is executed. Finally, the fifth step is the *Performance Evaluation Process*. During this phase the output of the execution is used to carry out a qualitative and quantitative evaluation of the risk-aware business process. The Re-Engineering Process phase is the one that makes business processes risk-aware. In this phase faults (here called "threats") are identified and modeled. If faults occur they impact the functionality of resources until one or more affected resources are no longer available. In the worst case a resource represents a single point of failure and consequently if it fails will hinder the execution of the related

process activity. If a threat is detected, an appropriate countermeasure process is invoked to counteract the threat. However, if this cannot be done, a recovery process can be invoked to re-establish the functionality of the affected resources until they are available again for the respective business process activity. On the basis of the ROPE methodology, a reference model for risk-aware BPM is proposed by Jakoubi et al. [JT09, JTGK10b]. The main limitation of this approach is that it does not provide support during the execution of a business process but only helps in the identification of risks and their possible mitigations at design time.

DI02: In the approach proposed by Sienou et al. [SLPK10, SLPK08, SLKP07, SLPK09, SLP08, SKP06, SKLP08, KSLP07, Sie09], an integrated framework, called Business Process - Risk Management - Integrated Method (BPRIM), combining the domain of risk management and business process management is proposed. In particular, in this approach, activities that are commonly undertaken during the design stage of a business process (such as process modelling and analysis) are systematically linked to those activities from the risk management lifecycle [Sta09] to produce an integrated BPM and risk management (RM) lifecycle. Furthermore, the relationships between the concepts commonly encountered in the field of BPM (such as activity and resource) and RM (such as risk event) are explicitly studied and modelled (using class diagrams). Finally, a set of graphical notations, based on the Event-driven Process Chains (EPC) language, that can be used to annotate business process models with risk-related information (such as risk factor, control, and stakeholder) is proposed. Despite models are annotated using a language formally defined, the approach does not support implementation and cannot be used during process execution.

DI03: Cope et al. [CKE+10, CKE09, CDE+10] propose an approach in which a number of risk-related modeling constructs are defined. These constructs are an extension of the Business Process Model and Notation (BPMN) [Obj08a] language. By applying these constructs, one can encode risk-related information in a process model, such as the various risk events that can occur and the mitigation actions that can be taken. Furthermore, this approach also introduces a state-change event notation such that the "causal chains of failure" of a resource can be captured. This approach is somewhat inspired by the Bayesian network analysis technique, and it is limited to design time analysis.

DI04: In the approach proposed by Weiß and Winkelmann [WW11], the Semantic Business Process Modelling Language (SBPML) [BTWW10] is extended with a number of risk-related constructs, expressed as a set of graphical notations, such that the inclusion of risk-related information (such as risk events, risk control actions, and risk type) into business process models can be achieved. This approach has been specifically developed in the financial domain. It addresses a type of risk commonly encountered for the financial industry, called "operational risk". The approach does not support implementation and cannot be used during process execution.

DI05: The work by Asnar and Giorgini [AG08] addresses business process risks in the context of business continuity management. Building upon the Tropos Goal-Risk Framework [AG06] (defined by the same authors) and the Time Dependency and Recovery Model [ZBES07], an extended goal-risk framework is proposed. This framework consists of three layers: the asset layer which consists of business process goals, activities, and business artefacts; the event layer which consists of various events (including risk events) that can impact the asset layer; and the treatment layer which consists of a set of risk treatment activities that can mitigate the impact of the occurrence of the risky events modeled in the event layer. Several risk analysis techniques based on this extended goal-risk framework are also proposed. These techniques are strongly influenced by existing risk analysis techniques (notably the Cost-Benefit analysis technique and the Treatment analysis technique), and are limited to design-time analysis.

DI06: In the approach proposed by Mock and Corvo [MC05], a number of risk-related constructs are proposed, which can be used to annotate an EPC process model with risk events. The severity of each risk event (also called the risk priority number) and the causal chains of risk events can also be modeled. To complement these constructs, this approach applies the failure modes and effects analysis (FMEA) method for risk analysis [US 49] to identify the risk events in a process and the propagation of those events, but it is limited to design-time analysis.

DI07: Rosemann and zur Muehlen [RM05] propose a taxonomy of risks related to business processes. They identify five risk types: goal risks, structural risks, data risks, information technology risks, and organizational risk. Goal risks identify risks that can compromise the achievement of process or activity objectives.

Structural risks represent errors committed during the process design phase which may prevent the process from achieving its goals. Data, information technology, and organizational risks are risks that can compromise the correct execution of a process by damaging integrity of data, reducing system availability or reducing the performance of a resource. These five types of risks can be represented using four risk model types: i) risk structure model describing the hierarchical relationships between risks; ii) risk goal model defining, with the use of matrices, possible occurrences that could lead to a situation where the goal is not achieved; iii) risk state model describing the dynamic characteristics of a risk through non-hierarchical interrelationships between risks and the causal relationships between risks and consequences; iv) Event-driven Process Chains (EPCs) extended with risks that can be assigned to individual process steps. An extension of this work is proposed in [NCMR06], where the authors describe a four-step approach to integrate risks in business processes at the operational and strategic levels via value-focused process engineering, despite this extension the approach is limited to design-time, as is the original approach.

DI08: In the proposal by Rotaru et al. [NCMR06, RWC+11, RWCN08], the Value-Focused Process Engineering (VFPE) model [KT98] (which is based on the extended e-EPC meta-model) is further extended to formalize the concept of risk within business process models. In particular, this approach attempts to provide a common syntax to represent risks in a goal-oriented business-process model [RWCN08]. Moreover the authors propose a utility calculation technique that can be used at design-time to determine optimal risk countermeasure solutions. However, the utility calculation technique proposed does not work for running process instances and requires a set of countermeasure solutions as input.

DI09: Betz et al. [BHO11] use XML Nets [LO03] (a variant of Petri Nets) to model risk-aware BPM systems. In particular, they define a risk construct as a risk event caused by resources. Such risk constructs are linked to activities of a process. Then, risk countermeasure tasks are explicitly modeled as sub-processes of the activities affected by the risk events. If there is more than one countermeasure activity that can be applied to address a particular risk event, several process models will be generated, each capturing a particular countermeasure activity. This approach then proposes a method based on simulation to choose the best process model variant based on process cost and flow time information. This approach provides some support for risk mitigation, but only at design-

time. Moreover it requires a predefined set of mitigations as it is not capable of discovering any by itself.

DI10: The approach proposed by Hermann and Herrmann [HH06] focuses on data security risks in BPM. A set of graphical notations representing the security requirements of business processes is proposed. These security requirements then guide the evaluation of the security risk of the business processes (in terms of the non-satisfaction of the requirements). If the security risk is higher than a predefined tolerance level, a set of risk mitigation activities are added at design-time to the model as risk treatment such that the process security risk can be reduced to an acceptable level. The limitation of this approach is that the risk mitigation support provided is a hard coded approach that only provides mitigations calculated at design-time.

DI11: Strecker et al. [SHF10] propose a multi perspective risk modeling method for an IT infrastructure based on the Multi-Perspective Enterprise Modelling (MEMO) Meta Modelling Language (MML) [Fra10]. In this approach, a risk modeling language (RiskML) is proposed, that can be used to express risk-related information (such as the risk events, risk countermeasure activities, and risk propagation) using the MEMO MML as the "conceptual foundation" [SHF10]. Risk-related information annotations can be added to existing organizational models at different levels of granularity (strategic level, business process/operational level, and IT/infrastructure level). Despite models are annotated using a language formally defined, the approach does not support implementation and cannot be used during process execution.

DI12: The approach proposed by Karagiannis et al. [KMS07] focuses on addressing the (non-)compliance risk of business processes to the Sarbannes-Oxley Act [10702] standard. This approach, specifically focused on the financial domain, introduces several risk-related constructs to capture risk-related information. Such constructs can then be used to annotate business process models with information related to risks. This approach also describes how risk annotation can be used for risk-informed business process design assisting the modification of the process model through the addition of control activities. A six-step framework that can be used to realize a risk-aware business process management system is proposed.

DI13: In the approach proposed by Taylor et al. [TGM08], a simulation environment is developed using the jBPM stack and the jBPM Process Definition Language (JPDL). Several risk-related constructs, such as key risk indicator (KRI), key performance indicator (KPI), and risk event, are proposed. These constructs are used to annotate process models with risk information. Both qualitative measurement and quantitative measurement of KPI and KRI are supported. Through the application of simulation and fuzzy logic, the effects of risk events on some pre-defined KPIs and KRIs are evaluated. Despite executable process models are used as part of this approach, it is not meant to cater for risk identification at execution time.

DI14: Panayiotou et al. [POAG10] propose an internal audit process as a method to perform risk assessment and identify relevant risk mitigation actions for virtual enterprise networks. In particular, this approach defines a technique to collect relevant information about processes in a structured manner such that it can be subsequently filtered and reported for the purpose of risk analysis. Relevant templates and tools (developed using the Sybase PowerDesigner enterprise modeling software) that can be used to aid the application of the proposed approach are provided. A number of risk-related constructs that can be used to annotate at design-time process models with risk-related information (such as risk mitigation activities) are also proposed.

DI15: In the approach proposed by Lambert et al. [LJJ06], the integrated definition (IDEF) language [IDE93] is used to model business processes and extended to include the concept of "source of risk" into business process models. Several simple examples demonstrating the use of the proposed construct are shown, but the syntax proposed is not formally defined and there is no implementation that supports the approach during process execution.

DI16: In the work by Bai et al. [BBK06, BPK07, BPK06], business process models are represented as graphs: nodes represent tasks and arcs represent gateways. A precedence matrix is also used to define the topology of process models. Process-related errors and error-mitigation activities are annotated in the corresponding error and control models proposed by this approach. These models are then used to reason about risks of processes. This approach makes use of optimization techniques to determine the best place(s) in the workflow graph to position relevant error mitigating tasks. This approach applies several existing

risk analysis techniques, such as the Conditional Value-at-Risk technique [RU00], to evaluate risks by taking into consideration the likelihood of the risk events modeled, their consequences, their propagation, as well as the risk mitigation activities applied. The ultimate goal of the design-time analysis is to ensure that an optimal placement of risk mitigation activities is achieved. The limitation of this approach is that the optimization placement of risk mitigations is executed at design-time, requires a predefined set of mitigations, and generates a predefined "mitigation" for every possible instance of a process model.

DN01: Bhuiyan et al. [BIK+07, IBKG09] proposed a technique to quantify the criticality and vulnerability of actors in a business process. This is achieved by analyzing the incoming and outgoing edges of actors in an actor dependency model represented using the i^* framework notations [Yu97]. The results of this analysis are then used to inform the design of the corresponding BPMN business process models in order to reduce/mitigate the negative consequences resulting from the failure/unavailability of critical actors. Therefore, risk-informed business process design is also proposed. This approach is developed mainly to address organizational risk using the taxonomy proposed in [CSI11]. Also in this case the risk mitigation support provided is limited to design-time, providing only predefined and static mitigation strategies.

DN02: Fenz et al. [FN09, FEN09, FE09, Fen10] propose a design-level approach based on a set of techniques that can be used to analyze risks of a business process, mainly considering risks from a resources (such as IT assets) point of view. This approach focuses on the consequence analysis of a risk event, its likelihood, its propagation, and the overall risk level of a business process. Such an analysis is carried out through the application of the "resource importance" calculation formula based on the structure of the Petri-net model in which the corresponding business process is depicted. The analysis of the occurrence probability of a risk event (and the propagation of risk events) is achieved through the application of a Bayesian network analysis. The development of the Bayesian network is based on the security ontology proposed by the same authors (expressed using the Web Ontology Language (OWL) [MH+04]). This ontology contains concepts related to information security and risk, such as vulnerabilities, threats, and countermeasures. The limitation of this approach is that the analysis is executed at design-time and no support during process execution is provided.

DN03: In the approach proposed by Kaegi et al. [KMZN06], a process model described in BPMN is simulated via an agent-based modeling technique to analyze business process-related risks. A risk estimation formula is also used in the process risk analysis. There is no evidence to suggest that this approach is prescribed for any particular type of risk or domain. As this approach is based on simulation it only provides support at design-time.

DN04: In the work by Bergholtz et al. [ABE+05, BGJ+05, SGD05], an approach to facilitate risk-informed business process design (driven by risk treatment activities) is detailed. This approach starts with a business model described using the Business Model Ontology (BMO) [Ost04] language. Such model is then transformed into a value-web model described using the e^3-value model notation [GYR06]. Then, through the aid of the corresponding activity-dependency diagram, the value-web model is transformed into a BPMN-based process model. At each stage of the model transformation, the approach suggests the determination of risk events that may occur in the model being studied, and the modification of the model such that relevant risk mitigation activities are integrated into the final (derived) business process model. In other words, the end product of such a process is a risk-informed business process model which already contains relevant risk mitigation activities. In this case the support provided for risk analysis and the risk mitigation is limited to design-time.

DN05: In the work by Jallow et al. [JMV+07], an approach to analyze risks in business processes is proposed. Given a set of identified risk events and their occurrence probabilities, Monte Carlo simulation [Met87] is applied in order to assess and quantify the impact/consequences of these risk events (in terms of time, cost, performance, and other objectives) on each process activity and on the overall process. As this is a simulation-based approach, it only provides support at design-time.

DN06: In the approach proposed by Singh et al. [SGD+08], a technique to evaluate a workflow's non-completion risk due to uncertain/dynamic information is proposed. The term 'non-monotonic predicate' is used to refer to such information. Examples of non-monotonic predicates include the number of injured passenger(s) in a car accident, or the status of traffic at the time of the accident. This information is not likely to be known until run-time. A method to quantify the confidence level of the non-monotonic predicates of a workflow

is also proposed. When the confidence level of a non-monotonic predicate of a workflow is below a certain threshold, the workflow is considered to be risky. In this situation, this approach suggests the use of a backup workflow such that the non-completion risk of the related workflow instance is mitigated. Finally, an approach to generate a backup workflow based on workflow execution history is also briefly described. It produces an additional branch in the process model that can be taken in case the normal execution turns out to be risky. The main limitation of this approach is that it only provides support at design-time and that a solution is hard-coded in a process model.

RT01: In the approach proposed by Kang et al. [KCK09], a technique to estimate the probability that a process instance enters an abnormal termination state is defined. Process-related historical data, containing information about the normal or abnormal termination of process instances, is used to inform the probability estimation calculation. Then, a run-time risk estimation algorithm is developed such that appropriate risk alerts can be produced when risky situations are detected. This approach does not seem to be prescribed for any specific type of risk and can be applied to any domain. However, it only looks at abnormal termination states, not at any other possible exception that may affect a process instance. Finally, the approach requires the generation of a Representative Pattern (RP) "defined as the comprehensive set of all events that a process instance in a process model should have until completion" [KCK09]. This requirement clearly cannot be satisfied in case of business process models containing loops.

PE01: In the work by Jans et al. [JDV11, JLV01, JWLV11], business process logs are analyzed, such that risks related to financial fraud can be identified and the occurrence probability of these risks can be estimated. In particular, the ProM tool is used to aid the reasoning about the inadequacy of internal controls and the estimation of the likelihood of a user subverting existing processes, such that transaction fraud can be committed. Interesting fraud-related properties, such as segregation of duty, were verified using the logs. This approach managed to uncover suspicious process instances that were not detected during traditional internal audit processes [JDV11].

PE02: In the approach proposed by Wickboldt et al. [WBL+11], historical information (such as logs) from business processes is annotated with a number of risk-related constructs, such that various types of risk analyses can be conducted.

	Process Design	**Process Implementation**	**Process Enactment**	**Process Diagnosis**
Risk Identification	DI01, DI02, DI03, DI04, DI05, DI06, DI07, DI08, DI09, DI10, DI11, DI12, DI13, DI14, DI15, DI16, DN01, DN02, DN03, DN04, DN05, DN06	DI01, DI09, DI12, DI13, DN03, DN04	-	PE01, PE02, PE03, PE04
Risk Monitoring	n.a.	n.a.	RT01	n.a.
Risk Prevention	-	-	-	-
Risk Mitigation	DI01, DI02, DI05, DI08, DI09, DI10, DI12, DI14, DN01, DN04, DN06	-	-	-

Table 2.2: Risk Management activities supported by each evaluated approach.

In particular, techniques to identify and evaluate risks in business processes (including risk probability estimation and impact analysis) are proposed.

PE03: Pika et al. [PAF+13a, PAF+13b] propose an approach for predicting overtime risks based on statistical analysis. Logs are analyzed to identify five process risk indicators whereby the occurrence of these indicators in a trace indicates the possibility of a delay. The main limitation of this approach is that they limit their scope to the identification of indicators of risks or of causes of faults. Moreover, these indicators do not consider the data perspective and have been designed to support overtime risks only.

PE04: Suriadi et al. [SOAH13] propose an approach for Root Cause Analysis based on classification algorithms. After enriching a log with information like workload, occurrence of delay and involvement of resources, they use decision trees to identify the causes of overtime faults. The cause of a fault is obtained as a disjunction of conjunctions of the enriching information. This approach suffers from two shortcomings. Firstly, the approach is limited to the identification of fault causes without proposing possible solutions in order to avoid such causes. Secondly, fault causes do not take in consideration the data prospective.

Table 2.2 provides a summary of the types of risk-related activities that the approaches described so far support at each stage of the BPM lifecycle. With

respect to the BPM lifecycle shown in Figure 2.2, all the above proposals only cover the phases of risk analysis and process modeling, except for the approach by Kang et al. that is specifically focused on run-time analysis, and a few approaches, out of our scope, that are focused on post-execution analysis. Among these approaches, none of them specifies how risk conditions can be concretely linked to run-time aspects of process models such as resource allocation, data variables and control-flow conditions. The few approaches that cover the process implementation phase do it only for simulation purposes. Given that none of these approaches deal with run-time aspects (i.e. execution) of business processes, the topics of risk monitoring, risk prediction and prevention, and risk mitigation are not yet explored.

2.3 Business Rules

Business Rules is a approach that uses rules to model business processes. The Business Rules approach formalizes an enterprise's critical business rules in a language that managers and technologists can understand (e.g. "It is obligatory that each product pass a quality test"). Business rules have been defined as a "declaration of policy or conditions that must be satisfied" [JO98], and create an unambiguous statement of how information is used in a business to make a decision.

Business rules can be represented using the same structure as rules in active databases [DBM88] using three basic elements: i) an *event* that specifies when the rule must be used; ii) a *condition* that must be verified before an action is executed; iii) an *action* that specifies what must be done. Rules that use this format are usually named ECA. Different extensions of the ECA (Event Condition Action) rules have been provided. For example, the ECAA rules provide the possibility of an alternative action if the condition is not verified, the EA rules are condition-less, and the EC^mA^n rules have multiple conditions and actions.

Business rules can also be used to model processes and workflows, e.g. in the work by Knolmayer et al. [KEP00]. The authors show how to model sequential actions, parallel actions, interaction of actions and other flow constructs. They also propose a way to model actors and data. Though business rules can be used to model business process, they also introduce disadvantages as, for example, the creation of a new process may be difficult because new rules may contrast with old rules. In order to provide support for business roles standards, such as the Semantics of Business Vocabulary and Business Rules (SBVR), have been defined

and both academic and commercial engines, such as DECLARE and the WebSphere ILOG JRules Business Rule Management System, have been produced.

The sensor architecture that will be presented in this research resembles similarity with business rules, despite it is not based on them.

The Semantics of Business Vocabulary and Business Rules (SBVR) [Obj08b] is the Object Management Group (OMG) implementation of business rules. It is an adopted standard that aims to define a "structured natural language" for capturing business processes. SBVR is an integral part of the OMG's Model Driven Architecture.

The SBVR defines the vocabulary and rules for documenting the semantics of business vocabularies, business facts, and business rules.

2.3.1 Academic business rules engines

Among the academic business rules engines relevant is DECLARE [PSA07]. It is a constraint-based BPMS developed by the Eindhoven University of Technology. It proposes a declarative constraint-based approach for process modeling and execution. The system is composed of three components [PSA09]: i) a *Designer* used for creating constraint templates, define organizational structures, and creating and verifying constraint models; ii) a *Framework* to execute and dynamically change models; iii) *Worklists* that allow user to access active instances and perform tasks.

In DECLARE, constraints are semantically specified using Linear Temporal Logic (LTL). These constraints can be created using constraint templates. There is no limit on the number of constraints that can be specified for a model. DECLARE supports two types of constraints: *mandatory* and *optional*. The system is realized so that it can support the execution of a process model and provide the opportunity to modify a process during its execution. Finally, the system automatically verifies if errors have been introduced after the modification of a model. The system can verify the presence of *dead activities*, *conflicting constraints*, and *history based errors*.

Though quite interesting as an approach, especially for the low-impact mitigation that the system provides, DECLARE is not particularly suitable for modeling large and complex business processes. The specification of a process may require a lot of constraints, that make process models difficult to read and that during their definition can easily introduce errors. Moreover, these constraints can only use events regarding the execution of activities and do not consider other information related to the execution of processes (e.g. variables, resources, etc.).

2.3.2 Commercial business rules engines

Among the commercial business rules engines is worth mentioning the WebSphere ILOG JRules Business Rule Management System [Sti09]. This tool, which we refer to as JRules, is a business rules management system developed by IBM.

JRules consists of three components allowing business users to create and execute business rules. These are:

- *eXecutable Object Model (XOM)*. This is the model that enables the execution of rules, which can be created from compiled Java classes or XML Schema.
- *Business Object Model (BOM)*, is an abstract object-oriented representation of the "information model" that defines the concepts of a given business. It contains a set of classes which the rules act up on, and it is mapped onto the XOM.
- *Business Rules*, are expressed in a structured natural language. For their definition the user can use business objects taken from the domain represented in the BOM and the vocabulary model (VOC).

2.4 Business Process Monitoring

Among practitioners and academics an emerging need is the possibility of monitoring the status of a business process execution. Business Process Monitoring has been the response to this need. In particular, two main streams can be identified in this area: i) Business Activity Monitoring (BAM), and ii) Complex Event Processing (CEP).

2.4.1 Business Activity Monitoring

The term BAM, originally coined by Gartner [Gas04], refers to real-time filtering, aggregation, analysis, and presentation of information about activities executed as part of a process. Generally, BAM focuses on providing a real-time summary of business activities to operations managers through the aid of dashboards containing key performance indicators (KPIs). BAM solutions are mainly proposed by commercial vendors. In the following we will present the Oracle Business Activity Monitoring (Oracle BAM) [Ora11], which is based on sensors like the approach presented in this thesis.

Oracle BAM is the BAM solution proposed by Oracle. It monitors the occurrence of specific events of a BPEL [AAB+05] process through the use of sensors (BPEL Process Manager Sensors). Several types of sensors can be defined on the basis of the context in which they operate. The types of sensors provided are: activity sensors, variable sensors, and fault sensors. *Activity sensors* monitor the execution of activities in a BPEL process. They can monitor the execution time of an activity or variables that the activity modifies. *Variable sensors* are used for monitoring the values of variables modified by a BPEL process. In this way the sensor can monitor the value of input and output data of a process. Finally, *fault sensors* can monitor BPEL faults.

Sensors operate at run-time during the execution of a process. When a sensor is triggered, a new sensor value (i.e. value monitored and time stamp) is created and all actions associated with the sensor are performed, such as storing the information retrieved in a database or publishing the information to external sources. The triggering of a sensor is dictated by an *Evaluation Time* specified during the definition of the sensor. Five evaluation times can be specified: i) on *Activation* means that the sensor fires just before the activity is executed; ii) on *Completion* means that the sensor fires just after the completion of the activity; iii) on *Fault* means that the sensor fires if a fault occurs during the execution of the activity; iv) on *Compensation* means that the sensor fires when the associated scope activity is compensated; and v) on *Retry* means that the sensor fires when the associated activity is retried.

An interesting point of this approach is the use of sensors to monitor the execution of a business process. In general, sensors are used in different areas [ONS96, Ché97, BW04], principally because they are light-weight and not expensive (in terms of resource consumption). In the context of Oracle BAM, sensors cannot monitor sophisticated conditions. It is impossible to define conditions incorporating information such as resource allocation strategies and order dependencies. Furthermore, sensors can only retrieve information specific to the current running instance, making impossible the definition of conditions that use historical data and information from other running process instances. Moreover, in this approach sensors can only be triggered by process events impeding the definition of a condition for monitoring deadline risks. These two limitations make this approach not a not suitable solution since we required a solution that could address any type of risk related to business processes.

2.4.2 Complex Event Processing (CEP)

Complex event processing (CEP) is an emerging technology for analyzing and controlling complex series of events. Events can be produced by different sources such as databases, email accounts as well as process engines. These sources generate simple and independent series of events, which using rules are aggregated to form relationships and patterns. These relations and patterns are detected by CEP engines. A CEP engine is a system capable of analyzing a stream events and identifying the significant ones. Generally, CEP engines use languages similar to Structured Query Language (SQL) for the definition of rules [WREW11]. The definition of a rule requires the specification of a sequence of events, the occurrence of which identifies the manifestation of a complex event. The condition of a complex event can also be defined in terms of attributes belonging to simple events. Moreover, a temporal window within which these events must happen can be specified.

Although a CEP engine is a generic tool, commercial vendors integrated it in their BPMSs, e.g. webMethods Business Events[1], ARIS Process Event Monitor [DB07], and SAP Sybase [Syb11]. The use of CEP engines for monitoring purposes has also been explored in academia [GPL+10, HSD10].

Gay et al. [GPL+10] propose the use of complex event processing for workflow monitoring. In their approach, based on Petri nets, they identify six events representing the basic activities that a workflow can perform (i.e. Transition activation, Resource allocation, Resource liberation, Advance token, Start workflow, and End workflow). Using these simple events they have created six complex events that represent unwanted situations: i) *Lack of resource*, this situation occurs when a transition activation event (i.e. and transition is ready to be fired) is detected and after the estimated time needed to process a token an advance token event (i.e. a transition produces a token in a place) is missing; ii) *Activity delay*, this situation occurs when within a specific time window (i.e. the estimated time needed to process a token) two advance token events are expected, but only the first one is detected; iii) *Lack of resource delay*, this situation occurs if a lack of resource and activity delay are detected; iv) *Transition delay*, this situation occurs if an activity delay is detected and a transition activation event is missing; v) *Workflow delay*, this situation occurs when within a specific time window (i.e. the estimated time needed to execute the entire process) a start workflow event and an end workflow event are expected, but only the first event is detected; and vi) *Interruption warning*, this situation occurs when within a specific time

[1] `http://www.softwareag.com/au/products/wm/events/overview/default.asp`.

window (i.e. the estimated time needed to execute the entire process) a complex event transition delay a transition activation event are expected, but only the first is detected. Finally, Gay et al. suggest an eventual mitigation that requires the invocation of some remedial actions when an unwanted situation is detected.

A different approach is proposed by Hermosillo et al. [HSD10], which is concerned about Quality of Service (QoS) for on-line applications. They propose a framework named CEVICHE (Complex Event processIng for Context-adaptive processes in pervasive and Heterogeneous Environments) for dynamic business process adaptation. Their solution uses the AO4BPEL (Aspect-Oriented extension to the Business Process Execution Language) framework [CM07], to provide the possibility of adding and changing a service. AO4BPEL integrates Aspect-Oriented Programming with BPEL creating a wrapper around the BPEL interpreter. To be able to add and change a service, AO4BPEL needs to know where the adaptation must be applied and what kind of adaptation must be executed. The information about the business process (BPEL), the adaptation condition (CEP rules), and the adaptation definition are stored inside a specific type of file (SBPL file). During the execution of a process the CEP engine will check the events produced filtering them using the rules defined. If the CEP engine detects a delay during process execution the system will automatically apply the adaptation specified in the SBPL file.

Though the use of a CEP system for risk monitoring can be an interesting solution, performance is a strict requirement for this research and CEP systems typically suffer from performance overheads which limit their applicability to real-time risk detection [WREW11]. Moreover a CEP system can only detect situations using event-driven triggers, and these cannot be used to detect a deadline risk. Finally, the solution by Hermosillo et al. [HSD10] is not adequate for risk monitoring. The main limitation of the first solution is the finite set of unwanted situations which this approach can detect. While, Guy et al. [GPL+10] suggest a possible future development of remedial actions to invoke as a consequence of a detection but do not implement it. The second approach, on the other hand, mainly focuses on mitigating QoS problems using predefined remedial actions.

2.5 Business Process Adaptation and Exception Handling

The achievement of effective risk mitigation requires dynamic changes on business process models at run-time. The safe application of changes to a running business process is not trivial and it is investigated in the area of Business Process Adaptation. Business Process Adaptation covers that branch of the BPM discipline that studies how to modify (usually at run-time) business process models without affecting negatively the execution of business process instances. In the area of Business Process Adaptation several approaches have been proposed, among which we can find adaptation patterns [WRRM08] and adaptation frameworks [RRD03, RRD04, MGR04, WWB04, KKK07, SMBH11].

The topic of adaptation patterns has been investigated in the work by Weber et al. [WRRM08]. They identify a set of 18 change patterns, which are commonly used to modify business process models. They subdivide the patterns into two groups: adaptation patterns and patterns for changes to predefined regions. The first group allows structural modifications to a process model using high-level change operations and contains 14 of the 18 patterns identified (e.g. Insert Process Fragment, Delete Process Fragment, Move Process Fragment). The second group contains patterns that allow participants to add information about unspecified parts of the process model.

In order for adaptation patterns to be supported adaptation frameworks are required. The most advanced framework for process adaptation is based on the work by Reichert et al. [RRD03, RRD04]. In their work, they propose a generic framework for supporting process model and process instance changes and criteria for correctly propagating process changes to executing instances. Using Well-Structured Marking-Nets (WSM-Net), they propose a set of propositions which can guarantee, under some pre-conditions, the prevention of deadlocks, missing input data and lost updates. The result of their research produced the ADEPT/AristaFlow system [DR09].

Various other frameworks have been proposed for the dynamic adaptation of process instances. Müller et al. [MGR04] propose the workflow management system AgentWork, which provides the ability to modify process instances by dropping and adding individual tasks based on events and rules. Weber et al. [WWB04] instead propose the workflow management system CBRFlow, which uses case-based reasoning to support run-time adaptation by allowing users to define business rules during process execution. Hermosillo et al. [HSD10] pro-

pose the framework CEVICHE, which is a service-based framework that uses the AO4BPEL (Aspect-Oriented for BPEL) language [CM07] to provide the option of skipping or reallocating tasks to other services in an ad-hoc manner.

Kim et al. [KKK07] propose an approach for dynamic business process management using process change patterns. In their approach they propose the use of CVS for the creation of different versions of a process based on the concepts of revisions and variants. This approach proposes to use a common version of the model until a variation point is reached, then the particular version of the model is loaded. Variation points are constructed using a combination of pattern blocks. The patterns identified have been classified in three main groups: i) *Activity Split*; ii) *Activity Merge*; iii) *Activity Extend/Delete*.

Finally, Schick et al. [SMBH11] propose an approach for run-time adaptation of business process models. In their approach they propose the creation of portions of process at run-time. Schick et al. extend the YAWL system with two types of tasks: observer tasks and generator tasks. Observer tasks identify a point in the business process model where external messages and data are investigated using *matching rules*. The results of the investigation are used by a generator task. Using some predefined *construction rules* a generator task composes a subprocess joining predefined specific activities named *bricklets*. Once the sub process is ready it is executed as replacement of the generator task.

When considering Business Process Adaptation a topic that is often mentioned is exception handling, since dealing with exceptions often requires dynamic changes in a process. In general, the term exception handling refers to the process of responding to the occurrence of an exception with the purpose of re-establishing the execution of the process. Managing exceptions raised by the system during a process execution is essential for the completion of a process but not always possible. Different approaches have been proposed in the literature. Interesting are the approaches proposed by Hagen and Alonso [HA00] and by Adams et al. [AHEA05].

Hagen and Alonso [HA00] propose the integration of transaction concepts into exception handlers. In their approach, the transaction concepts provide a partial backward recovery (e.g. compensation and holistic back-out) while the exception handlers guarantee forward progress.

Adams et al. [AHEA05] propose dynamic exception handling through exlets. In their work they see an exception as a natural deviation from the normal work plan. To manage these deviations they propose the use of exlets. An exlet is a complete workflow process that can be automatically loaded by the system to

manage an exception. The selection of which exlet is necessary to load is achieved through the use of modified *Ripple Down Rules* [CJ88].

Finally, in the areas of Business Process Adaptation and Exception Handling, we can find approaches based on the use of planners [RMBCO07, MRM12, BKB+14]. These approaches use planners to automatically generate process models, which are used to mitigate instances for which an exception occurred. In order to generate a process model a planner requires knowledge about a goal (a set of conditions that should be fulfilled) that should be reached at the end of the process, and a set of activities that could be used as part of the process. For each activity it also requires a set of preconditions that need to be satisfied in order for the activity to be executed and a set of post-conditions resulting from the execution of the activity.

While the approaches discussed in this section could be used for risk mitigation purposes, they do not provide any help with the identification of which particular mitigation actions to use. On the other hand they can be used to provide support for the actuation of the mitigation actions that our approach will identify. In particular, for the purpose of this research, the approach proposed by Reichert et al. [RRD03, RRD04] is interesting because it provides a starting point to verify if the mitigation detected can be applied to the process. A problem can be the use of WSM-Nets proposed by the authors, as they have limitations imposed on their structure [RD98]. The approach proposed by Kim et al. [KKK07] does not really fit within the context of this research because a mitigation is not predefined and the use of a CVS is useless since a mitigation is specific for a specific instance in execution. The approach proposed by Schick et al. [SMBH11] may be of interest but a limitation of this approach is the use of constructor tasks which require predetermined knowledge of the remedial actions needed for mitigation. The approach of Hagen and Alonso [HA00] is not suitable in our context since it proposes a static handling of exception that requires to specify in advance where and how exceptions should be managed. The approach of Adams et al. [AHEA05] is not suitable for our purpose due to the necessity of predefining the point (i.e. the activity) in which the exception must be handled. Finally, the use of planners is not suitable since they require a predefined set of activities, for example a set of mitigation activities that is used for the generation of a new process, moreover we are not interested in producing a new business process. What we are interested in is to bring a process instance with an eventuated risk to a state where the risk is under control in order to allow the proper completion of the instance.

2.6 Business Process Improvement

The mitigation of risks is a problem requiring a solution which is often specific to the process instances for which risks eventuate. Moreover, in a large number of cases it is a run-time problem. This does not prevent the permanent mitigation of (specific) risks as a result of a process restructuring exercise. Restructuring processes using a systematic approach with the aim of increasing process performance is the goal of Business Process Improvement (BPI). This improvement is obtained by defining strategic goals and purposes of the organization, determining the expectations of an organization's stakeholders, and modifying business processes to meet goals and expectations.

In general, a BPI initiative is subdivided in four phases [SM09]: i) *framing the process of interest*, aiming to develop an overall process map and to identify a set of related processes, the target process should be identified and scoped identifying performance objectives of the process; ii) *understanding the "As-Is" process*, aiming to understand the process and why its performance objectives are not being met; iii) *designing the new "To-Be" Process*, aiming to design the to-be process as consequence of a process optimization analysis; iv) *implementing the new process*, aiming to create the new process design in the previous phase.

Among the different approaches proposed for business process improvement we are going to analyze PrICE. We decided to describe PrICE because it is the only one that has tool support.

PrICE The PrICE approach [Net10] is one of the approaches proposed for Business Process Improvement. This approach focuses its attention on the third of the four phases previously described, providing automated support for stages of the third phase that do not require direct human interaction.

The PrICE approach consists of four steps: i) *Finding applicable redesign operations*; ii) *Selecting suitable process parts*; iii) *Creating alternative models*; iv) *Evaluating performance of alternatives*. The first step tries to find possible redesign operations that are applicable to the process which we want to improve. These operations can be guided by *process measures* that provide a global view of the characteristics of the process and may reveal its weaknesses, or *process mining* which may reveal performance issues and possible improvements (e.g. bottlenecks) through the analysis of logs. Possible redesign operations are: parallelize, sequentialize, add task, remove task, group, compose, unfold. The second step requires the selection of process parts (i.e. components) for the application of the identified redesign operations. During this phase possible errors can be

produced due to the modification of the behavior. The approach for this step provides a tool to help users during the selection of parts of the process that fulfill the requirements of the identified redesign operations. The third step is the creation of alternative process models. This phase is composed of two sub-phases: "*apply redesign operation on selected process part*" and "*update model with alternative process part*" [Net10]. Finally, the last phase is the *evaluate performance of alternatives*. In this phase the performance of alternative models is evaluated. The evaluation is carried out through the support of the PrICE toolkit, in particular the performance of alternative models is obtained through simulation and measured using different performance dimensions.

Despite the clear link between BPI and risk mitigation, BPI approaches cannot be used to mitigate risks. In general, BPI approaches, such as the PrICE approach, require a high level of interaction with human users while we aim to automatically find the changes that are required to mitigate (thus improve) running process instances.

2.7 Business Intelligence

Supporting risk-informed decision making is a form of business process intelligence. In the literature a generally accepted definition for business intelligence is missing [PH03]. Different authors with the term business intelligence often refer to techniques and tools to support business decision-making.

As a consequence of the lack of a definition for the term business intelligence, different models for the business intelligence lifecycle have been proposed. In 1985, Gilad and Gilad [GG85] proposed a lifecycle composed of five phases: i) *collection*, in this phase information is gathered; ii) *evaluation*, in this phase the information collected in the previous phase is filtered; iii) *storage*, in this phase the information filtered is stored for future access; iv) *analysis*, in this phase the information is analyzed and interpreted; v) *dissemination*, in this phase the information is used as input for the decision-making process.

In the following we analyze business intelligence in the context of business processes.

Business Process Intelligence

The application of techniques of business intelligence to business processes is referred to as business process intelligence [CA09]. Usually, business process management systems produce a large number of event logs during process executions.

Logs are not only useful for recording events, but they can be a source of important insights into business processes. For example, these logs through adequate cleaning, filtering, aggregation, and analysis can be used to improve the business processes involved or predict undesirable situations.

Business process intelligence uses process information to provide support for: problems detection (e.g. bottlenecks), decision-making, and business process performance improvement. This functionality is provided by the aggregation of tools and techniques from different application areas [CA09]. These areas are:

- *Process Analysis*, refers to the analysis (using process mining techniques) of past and current process executions for the creation of explanatory, prognosis and decision models used for process optimization and prediction of critical situations;
- *Process Discovery*, refers to the analysis (using process mining techniques) of event logs for the detection of process, control, data, organizational and social structures;
- *Process Monitoring*, refers to the monitoring (using business process monitoring techniques) of running process instances for the detection of undesirable situations;
- *Conformance Checking*, refers to the analysis (using business process mining techniques) of a log to verify how close the behavior recorded in the log matches the behavior described in a process model;
- *Operational Support*, refers to the analysis of current and historical execution data, with the aim to predict future states of a running process instance, and provide recommendations to guide the user in selecting the next activity to execute based on certain objectives.

An example of a business process intelligence application providing process monitoring and operational support can be found in the work of Casati et al. [CDS02, CCDS04, GCC+04]. In their work they propose an architecture that supports the management of process execution quality via analysis, monitoring, and prediction of business process executions. The architecture uses an ETL (Extract, Transform, Load) application to collect data from several applications. This data is then stored in a data warehouse in order to be used to compute business metrics (i.e. measurable properties that can be associated with the data), e.g. the predicted completion delay of an activity. These metrics can be used as input to

condition-action rules to monitor the execution of a business process. Finally, predictive capabilities, supported by decision tree classification, are used to predict the most likely activity to be executed next in the business process [GCC+04], or to predict the most likely value of metrics of interest for business analysts, on the basis of historical data [CCDS04].

The approach of Casati et al. [CDS02, CCDS04, GCC+04] is thus complementary to our approach since it deals with process performance metrics, while our focus is on process risks. That said, their approach cannot easily be adapted to cater for our needs since predictions are based on most frequent execution paths, while we aim to suggest what activity to execute next on the basis of all options available, in order to reduce process-related risks. Further, the approach by Casati et al. does not offer mitigation capabilities.

Business Process Mining

Business process mining (also named Process Mining) is a relatively new area, which focuses on the use of event data logs to extract process related information. Process mining is a discipline that finds a place between the machine learning and data mining disciplines on the one hand and process modeling and analysis disciplines on the other hand [Aal11]. Process mining works under the assumption that it is possible to record events such that they refer to an activity, a case, a performer (often referred as originator), and a relative time.

The idea behind process mining is to discover, monitor, and improve business processes using knowledge obtained from event logs of processes. In process mining, event logs can be used for three main purposes [Aal11]: i) *process discovery*, which focuses on the automatic generation of a process model using a log; ii) *process conformance*, which verifies if an event log of a given business process conforms to a given process model of the same business process or vice versa; and iii) *process enhancement*, which focuses on the extension and improvement of business process models using information collected from log of the process. Two types of process enhancements can be identified [Aal11]: i) *repair*, which modifies the model to provide a better representation of reality; ii) *extension*, that adds new perspectives to the process model. The perspectives that can be used to visualize a business process model using process mining are:

- *control flow perspective*: which focuses on process control flow i.e. the ordering of activities

- *organizational perspective*: which focuses on resources (e.g. people and applications) involved in activities and on how they interact
- *case perspective*: which focuses on properties that characterize a process instance
- *time perspective*: which focuses on timing and frequency of events
- *data perspective*: which focuses on data i.e. the information produced and exchanged during the execution of activities

ProM The research conducted in the area of process mining led to the development of the ProM tool. ProM [ARW+07] is an extensible framework that supports a wide variety of process mining techniques. It has been obtained by the integration of EMiT [AD02], Thumb [WA03], and MiSoN [AS04]. The ProM framework has been developed as an open-source and pluggable environment which can be extended with new plug-ins.

The architecture of ProM supports five types of plug-ins: i) *Mining plug-ins*, used to introduce mining algorithms; ii) *Export plug-ins*, used to provide the possibility of exporting objects (e.g. graph models, Petri nets, EPCs, etc.); iii) *Import plug-ins*, used to import objects in the system (e.g. logs, business process models, etc.); iv) *Analysis plug-ins*, used to run analysis algorithms on mining results; v) *Conversion plug-ins*, used to convert between different data format (e.g. from EPCs to Petri nets).

Compliance Monitoring

When we talk about conformance in process mining, we also cover the area of process compliance, i.e. if a log complies with certain rules. Compliance monitoring addresses the problem of monitoring business process execution to detect if one or more compliance rules are violated. Several approaches have been proposed to address this problem [BDL+10, WZM+11, MMWA11, LRMKD11, MMA12]. Birukou et al. [BDL+10] propose an approach based on CEP for the monitoring of compliance violations in the context of service-oriented architectures. Tha approach of Weidlich et al. [WZM+11] also uses CEP to monitor the violation of constraint in business processes using CEP. These two approaches suffer from the typical limitations of CEP monitoring discussed in Section 2.4. Maggi et al. [MMWA11, MMA12] proposes an approach based on linear temporal logic (LTL) and business rules. Finally, Ly et al. [LRMKD11] propose an approach

based on compliance rule graphs. All this approaches suffer from the same limitation which is that they identify a violation only after it occurs.

Predictive Monitoring

Among the areas targeted by business process intelligence we mentioned operational support. Predictive Monitoring is a specific and relatively new type of operational support where the goal is preventing the violation of business constraints. Several approaches for predictive monitoring have been proposed [MDDG14, CSC+05, ASS11, FGP12, RSW13].

In the approach proposed by Maggi et al. [MDDG14], users can define business goals, represented in the form of LTL constraints, at any time during the execution of a process instance. These business goals are then used in combination with data logs to generate decision trees that provide a prediction on the possible outcome of the process instance.

Castellanos et al. [CSC+05] propose a platform for business operations management. This platform, equipped with various time series forecasting functionalities, allows the prediction of metric values and aggregated metric values, such as the number of orders expected for the next week.

Other predictive monitoring techniques focus on temporal aspects. Van der Aalst et al. [ASS11] propose an approach for cycle time prediction. The algorithm predicts the remaining cycle time of a process using an annotated transition system (a graph where each node represents a sequence of activities). Whenever the algorithm needs to predict the remaining cycle time of a process, it looks at the node representing the sequence of activities executed until now and returns the average of the remaining cycle times present in that node.

The approach of Van der Aalst et al. [ASS11] has been extended by Folino et al. [FGP12]. They propose a predictive clustering approach for cycle time prediction. In their approach context-related execution scenarios are clustered, and for each cluster a state-aware time prediction model similar to the model proposed in the approach of Van der Aalst et al. [ASS11] is built.

Finally, Rogge-Solti and Weske [RSW13] propose an approach for predicting the remaining cycle time of a process instance using stochastic Petri nets.

In general, predictive monitoring techniques, if adapted adequately, will provide valid solutions for predicting risk eventuation. The main limitation of the solutions presented is that they focus on temporal aspects, except for the approach of Maggi et al. [MDDG14]. The latter approach is not limited to temporal aspects but it allows the definition of only one constraint per business process, and

multiple risks can thus not be captured.

2.8 Summary

In this chapter we presented literature related to risk-aware business process management. In particular, we started with a brief introduction of business process management, putting some emphasis on its lifecycle [AHW03] and the worlflow reference model [Hol95].

We then introduced the concept of risk management focusing on some of the main risk analysis techniques and methods. In particular, we focused on approaches for process-related risk management in which we identified 27 approaches. We analyzed and classified them using as criteria the phases of the BPM lifecycle in which they operate and the types of risk management activity they support. The result of our analysis shows that there are no approaches which provide risk-related extensions of a process model that can be executed and used to monitor risk or to influence behavior of processes at run-time. Moreover, only two approaches perform run-time analysis [KCK09, SGD+08], but none of these can be used for risk mitigation or risk prevention.

We also investigated the concept of business rules and business process monitoring in order to understand if they could be used for risk monitoring purposes. The result of our investigation shows that the techniques available are not suitable either because they are not integrated with a BPMS or because they are too generic and suffer from performance overhead.

Approaches in the area of business process adaptation and exception handling were investigated to identify the best approach for applying dynamic changes to a business process model as result of a mitigation. Despite the variety of approaches, none of them is suitable for our purpose mainly because predetermined knowledge of the mitigation is required.

Finally, we investigated the areas of business process improvement and business (process) intelligence for possible applications for risk detection, mitigation, and prediction. In the area of business process improvement we considered the PrICE [Net10] approach. This approach cannot be used for our research because it requires a high level of interaction with human users while we aim to automatically find the changes that are required to mitigate (thus improve) running process instances. Business process intelligence, and in particular process mining, offers several approaches that may prove useful for the realization of risk detection, mitigation, and prediction techniques.

In conclusion, although there exist several approaches for risk management in the context of business process management, three main shortcomings can be identified: (i) lack of adequate support for the detection of the eventuation of process-related risks during process execution; (ii) lack of support for the prediction of such risks and for their prevention; and (iii) lack of support for their mitigation. In the next chapters of this thesis we present a framework for risk-aware business process management which addresses these shortcomings. The subsequent chapter introduces some fundamental concepts that are used several times in the following chapters.

Chapter 3

Formal Foundations

This chapter introduces some notions and formalisms which will are in the following chapters in order to elaborate about the approach proposed. In particular Section 3.1 discusses the concept of YAWL Specification, Section 3.2 defines the concept of event log, and Section 3.3 discusses optimization techniques such as linear programming and simulated annealing.

3.1 YAWL - Yet Another Workflow Language

In 1999 the Workflow Pattern Initiative was created with the intent of identifying the core functionality required for BPMNs, and from its creation over 100 patterns were identified. Among all modeling languages available at that time, none of them was able to provide a comprehensive operationalization of those patterns. In order to address this gap, Ter Hofstede et al. [HAAR10] proposed YAWL (Yet Another Workflow Language), a rich modeling language based on workflow nets (a subclass of Petri nets).

One of our solution criteria requires that our approach should be independent from a specific language or BPMS. At the same time we require that our solution should be executable. In order to address these two criteria we decided to use YAWL and its system as reference modeling language and BPMS along the excursus of this thesis. We decided to use YAWL since it is one of the most advanced modeling languages (from a point of view of support toward the workflow patterns) and it has been formally defined. This choice should not be interpreted as a limitation since it will not prevent the use of our approach with different modeling languages and BPMSs.

This section provides a simplified version of the YAWL specification, stripped of any information not relevant in the context of this research. For the full

definition of a YAWL specification we refer elsewhere [HAAR10]. A YAWL Specification is a set of YAWL nets that form a hierarchical graph structure, with a root net and zero or more subnets.

Definition 1 (YAWL Specification)**.** *A YAWL specification is a tuple = (NetID, ProcessID, NYmap) such that:*

- *NetID is the set of net identifiers;*
- *ProcessID is the process identifier of the root net, ProcessID ∈ NetID;*
- *NYmap : NetID → YAWLnet is a function linking a net identifier to a YAWLnet.*

A *YAWLnet* is a directed graph where nodes are conditions (represented as circles) and tasks (represented as boxes). Conditions represent states of execution, for example preceding or following the execution of a task. In particular, in a YAWLnet we can find two special conditions. These conditions are the input condition, representing the initial state of a net, and the output condition, representing the final state of a net. Each *YAWLnet* has only one input condition and one output condition. Tasks represent activities that need to be executed as part of a business process, that can either be automated or manual. Nodes in a *YAWLnet* are connected via arcs. Tasks can be connected either to tasks or conditions, while conditions can only be connected to tasks. Tasks may have join or split behaviors. In YAWL, choices based on data are captured by an XOR-split, if only one outgoing arc can be taken, or by an OR-split when one or more outgoing arcs can be taken. Similarly, we have XOR-joins and OR-join that merge multiple incoming arcs into one. YAWL also provides support for AND-split and AND-join, which are used when all outgoing arcs need to be taken (AND-split) or all input arcs needs to be synchronized (AND-join). Finally, tasks can only be executed by an authorized group of resources. A resource may also have special privileges on a task that he executes such as, suspending it or skipping it.

Definition 2 (YAWL Net)**.** *Let T and C be the set of tasks and conditions, respectively. Let V be a set of variables. Let U be the set of values that can be assigned to variables. Let R be the set of resources. Formally, a YAWL net is a tuple $N = (T_N, C_N, i, o, F_N, R_N, V_N, U_N, can_N, Split, Join, UserTaskPriv)$ where:*

- *$T_N \subseteq T$ is the set of tasks in N;*

- $C_N \subseteq C$ *is the set of conditions in* N*;*
- $i \in C_N$ *is the input condition;*
- $o \in C_N$ *is the output condition;*
- *A flow relation* $F_N \subseteq (C_N \setminus \{o\} \times T_N) \cup (T_N \times C_N \setminus \{i\}) \cup (T_N \times T_N)$*;*
- $R_N \subseteq R$ *is the set of resources authorized to perform any tasks in* T_N*;*
- $V_N \subseteq V$ *is the set of variables that are defined in the net;*
- $U_N \subseteq U$ *is the set of values that can be assigned to variables;*
- $can : R_N \rightarrow 2^{T_N}$ *is a function that associates with each resource the tasks that the resource is authorized to perform;*
- $Split : T_N \nrightarrow \{AND, XOR, OR\}$*;*
- $Join : T_N \nrightarrow \{AND, XOR, OR\}$*.*
- $UserTaskPriv : R_N \times T_N \nrightarrow 2^{Priviliges}$ *is a function that associates each resource and task with the set of task privileges the resource is authorized to perform on the task. Where Priviliges is the set of possible privileges, defined as* $Priviliges = \{Suspend, ReallocateStateless, ReallocateStateful, Deallocate, Delegate, Skip, Pile\}$*.*

We use the following auxiliary functions [HAAR10]. The pre-set of x, i.e. the set of input tasks or conditions, is defined as $\bullet x = \{y \in C \cup T \mid (y, x) \in F\}$. Similarly, we have the post-set of x, i.e. the set of output tasks, defined as $x\bullet = \{y \in C \cup T \mid (x, y) \in F\}$.

Following the convention of Ter Hofstede et al. [HAAR10], we write e.g. T_N to access the tasks of net N. Moreover, for a YAWL specification y, T_y is the set of tasks that occur in any of its nets, i.e. $T_y = \cup_{N \in NetID} T_N$, and for a set of YAWL specifications Y, T_Y is the set of tasks that occur in any of the nets of any of the specifications, i.e. $T_Y = \cup_{y \in Y} T_y$.

In our context we only have one Organizational model [HAAR10] and what is relevant for us is the set of resources R, to whom work items can be assigned. Finally, we define the set of *skippable* tasks as $\{t \in T_Y \mid \exists r \in R[skip \in UserTaskPriv(r, t)]\}$.

When executed, a process uses a YAWL specification as a reference. In particular, YAWL Tasks are considered to be descriptions of a piece of work that

forms part of the overall process. Thus, control-flow, data, and resourcing specifications are all defined with reference to tasks at design-time. At run-time, each task acts as a template for the instantiation of one or more work items. A work item $w = (ta, id)$ is the run-time instantiation of a task ta for a process instance id.

A new process instance id is started and initialized by placing a token in the input condition of a YAWL net. The token represents the thread of control and flows through the net as work items are executed. The execution of a work item (ta, id) consumes one token from some of ta's input conditions (depending on the task's type of join) and produces one token in some of ta's output conditions (depending on the task's type of split). In YAWL, work items are performed by either process participants (*user tasks*) or software services (*automated tasks*).

In the definition we developed, several simplifications were applied. First of all, we simplified the resource prospective. In the original specification, tasks generally are not directly associated with resources, but resources can be selected based on their roles or capabilities. In our case this simplification is possible since at runtime the YAWL engine automatically assign tasks to eligible resources. In the original specification, tasks can be executed as multi-instance, in our case this possibility has been removed. We also simplified the data prospective. In YAWL data are passed across tasks through an advance mapping of tasks variables to net variables and the other way around, since for the work presented in this thesis this information is not relevant we decided to leave it out.

3.2 Event Log

The introduction of PAISs allowed the automated execution of business processes, and with it the possibility of logging the execution of such business processes. These event logs allowed the emergence of fields like process mining. Event logs need to contain a minimum set of information in order to be used successfully for process mining purposes. In particular, each event in a event log should be linked to a specific process activity and to a specific process instance. Moreover all events should be ordered. In general, additional information is also provided. This is the timestamp of the event, the resource that was involved with the execution of the activity, and the data produced during the execution of the activity.

In this section will be presented the concept of *event log*. An event log is composed of several *traces* that are sequences of *events*.

Definition 3 (Event)**.** *Let V be a set of variables. Let U be the set of values*

that can be assigned to variables. Let T be the set of tasks that can be potentially executed during the execution of the business process. Let R be the set of resources that are potentially involved during the execution of the business process. Let $\mathcal{T} \subseteq \mathbb{N}$ be the universe of timestamps. Let Φ be the set of all partial functions $V \nrightarrow U$ that define an assignment of values to a subset of variables in V. An event e is a tuple $= (t, r, d, \phi)$, where:

- *$t \in T$ is a task ;*
- *$r \in R$ is a resource;*
- *$d \in \mathcal{T} \subseteq \mathbb{N}$ is a timestamps;*
- *Φ is the set of all partial functions $V \nrightarrow U$.*

Definition 4 (Trace)**.** *A trace $\mathbb{T}$ (a.k.a. process instance, or case instance) is a sequence of events. In other words, $\mathbb{T} \in \mathcal{E}^*$.*

Definition 5 (Event Log)**.** *An event log $\mathcal{L}$ is a multiset of traces where each trace (a.k.a. process instance) is a sequence of events, i.e. $\mathcal{L} \in \mathcal{B}(\mathbb{T})$.*[1]

In general, BPMNs, like for example the YAWL system, during the execution of business processes are capable of storing more information and their logs are richer and more complex. In the context of the YAWL system additional information is contained into a log, e.g. case id, work item associated with a given event, or type of event. In the following we will present the definition of a YAWL log that can easily be represented as an event log when this additional information is removed.

Definition 6 (YAWL Log)**.** *In the context of a set of YAWL specifications Y, with associated set of tasks T_Y and a set of root nets $\mathcal{R}$, a **YAWL log** is an extension of an event log $\mathcal{L}$ defined as $L = (\mathcal{E}, \mathcal{W}, \mathcal{C}, Model, WI, Case, Task, EvType, Time, Res, Inp, Outp)$ where:*

- *$\mathcal{E}$ is the set of events,*
- *$\mathcal{W}$ is the set of work items,*
- *$\mathcal{C}$ is the set of case identifiers,*
- *$Model : \mathcal{C} \to \mathcal{R}$ is a function relating cases to the root nets of the associated YAWL specification,*

[1] $\mathcal{B}(X)$ is the set of all multisets over X.

- $WI : \mathcal{E} \to \mathcal{W}$ *is a surjective function relating events to work items,*
- $Case : \mathcal{E} \to \mathcal{C}$ *is a surjective function relating events to cases,*
- $Task : \mathcal{W} \to T_Y$ *is a function relating work items to tasks,*
- $EvType : \mathcal{E} \to StatusType$ *is a function relating events to work item statuses,*
- $Time : \mathcal{E} \to \mathcal{T}$ *is an injective function relating events to timestamps, hence no two events in the log can have identical timestamps,*
- $Res : \mathcal{E} \to 2^{UserID}$ *is a function relating events to sets of resources, as some events may concern multiple resources (e.g. a work item being offered),*
- $Inp : \mathcal{E} \times Var \rightarrowtail \Omega$ *is a partial function relating events and variables to (input) values,*
- $Outp : \mathcal{E} \times Var \rightarrowtail \Omega$ *is a partial function relating events and variables to (output) values.*

3.3 Optimization Algorithms

Optimization algorithms are a family of algorithms specialized in the resolution of optimization problems. An optimization problem can be summarized as the problem of minimizing an objective function without violating certain constraints. Optimization algorithms constitute a core element in risk mitigation. For a risk associated with a business process there can be a high, potentially infinite, number of ways of mitigating it. It appears clear that mitigating a risk requires not only the identification of a mitigation but also the identification of the best and most effective mitigation.

In the literature we can find several studies and approaches for the resolution of optimization problems. For example:

- *convex programming* studies the case when the objective function and the constraint set are convex (a function is defined convex if its epigraph[2] is a convex set[3]);
- *nonlinear programming* studies the case where objective function and/or constraints contain nonlinear parts;

[2]The epigraph of a function is the set of all points lying on or above the graph of the function.
[3]A convex set is a set containing all line segments between each pair of its points.

- *stochastic programming* studies the case where constraints depend on random variables;
- *combinatorial optimization* studies the case where the set of solutions is or can be reduced to a discrete set;
- *heuristics and metaheuristics* study the case where no assumptions about the problem are made.

In the context of this research we will utilize two types of approaches for the resolution of optimization problems, specifically *convex programming* and *meta heuristics*.

3.3.1 Linear Programming

A mathematical optimization problem is the problem of finding the optimal value (minimum value) for a given function under a set of given constraints. It has the form:

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq b_i, \quad i = 1, \ldots, m. \end{array}$$

Here $f_0(x)$ is the *optimization function*, i.e. the function that we are interested in optimizing, and where $f_i(x)$, with $i = 1, \ldots, m$, are the *constraints*, i.e. the set of functions that define the constraints that a solution should satisfy in order to be admissible.

A *mathematical optimization problem*, is called a *convex optimization problem* if its optimization function and all its constraints are convex functions. A convex optimization problem having a linear optimization function and linear constraints, is called a *linear optimization problem* or a *linear programming problem*. A linear programming problem has the following (canonical) form:

$$\begin{array}{lll} \text{minimize} & \sum_{j=1}^{n} c_j \cdot x_j & \\ \text{subject to} & \sum_{j=1}^{n} a_{i,j} \cdot x_j \leq b_i, & i = 1, \ldots, m. \\ & x_j > 0, & j = 1, \ldots, n. \end{array}$$

It is also common to see a linear programming problem represented using matrices as:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax \leq b \\ & x \succeq 0 \end{array}$$

Here $\succeq$ represents component wise comparison, $c \in \mathbb{R}^n$ is the cost vector and c^T is the transpose of c, $x \in \mathbb{R}^n$ is the vector of variables, $A \in \mathbb{R}^{m*n}$ is the *coefficient matrix*, and $b \in \mathbb{R}^m$ is the vector of coefficients or *righthand side*:

$$c = \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix} x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} A = \begin{bmatrix} a_{1,1} & \dots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \dots & a_{m,n} \end{bmatrix} b = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

It is important to realize that each constraint is a half-plane, and all together they create a convex polytope[4] that is the space of admissible solutions. Analyzing the space of admissible solutions it is possible to derive if the problem admits a solution or not. If the created polytope is empty, then the linear programming problem does not admit a solution, if it is bounded then the linear programming problem has one or infinitely many solutions, finally if the polytope is unbounded the problem may have one or infinitely many solutions or be unbounded[5] based on the direction of the optimization function.

Solving a linear programming problem is not a trivial operation and several methods have been discovered during the years. The most famous is Dantzig's simplex method [Dan51]. The method proposed by Dantzig uses the polytope of the possible solutions, moving along its vertexes toward the vertex with the lowest value for the optimization function.

An *integer linear programming problem* is a type of mathematical problem. It is similar to a linear programming problem but has an additional constraint, all its variables must be integers, and it has the form:

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ & x \succeq 0 \\ & x \in \mathbb{Z}^n \end{aligned}$$

This additional constraint drastically increases the complexity of the problem, since in this case the space of admissible solutions is not a polytope anymore. The absence of a polytope makes the Dantzig's simplex method ineffective. A well known algorithm to solve an integer linear programming problem is the Branch and Cut algorithm [PR91]. The algorithm is a hybrid between the Branch and Bound algorithm [LD60] and the Cutting Planes algorithm [Gom58]. It first

[4] A polytope is a geometric object with flat sides. For example a polygon is a polytope in two dimensions.

[5] A linear program is unbounded if it is feasible but its objective function is infinite, i.e. it can be made arbitrarily "good".

drops the constraint requiring that all variables must be integers, obtaining the linear programming relaxation of the problem. If a solution is found and it is not an integer solution, the algorithm splits the problem into two subproblems (branches), by choosing a variable a of value j that does not have an integer value.

The first subproblem is the original problem with an additional constraint that requires a to be less than or equal to $\lfloor j \rfloor$. The second subproblem requires as additional constraint a to be greater than or equal to $\lceil j \rceil$. The algorithm then computes the linear programming relaxation of both the problems and keeps going until the optimal solution is found. At each step the best integer solution is saved, and in case the value of the optimal solution for a linear programming relaxation is not better than the optimal solution found so far the entire branch is discarded since the optimal solution to the original problem cannot be found in that branch.

Finally, a less restrictive version of an integer linear programming problem, is a *mixed integer linear programming problem*, where at least one variable must be integer.

3.3.2 Simulated Annealing

In mathematical optimization and in computer science, metaheuristics are general algorithmic frameworks designed to find a solution for complex optimization problems [BDGG09]. Metaheuristics are capable of providing a sufficiently good solution to optimization problems and are often used in contexts where incomplete or imperfect information is provided or limited computation capacity is available. Metaheuristics can be used for a variety of problems since they make few assumptions about the optimization problem that will be solved. Metaheuristics compared to optimization algorithms and iterative methods do not guarantee that the global optimal solution will be found.

Several algorithms fall under the umbrella term *metaheuristics*. Since the purpose of this chapter is to provide an understanding of the techniques used in this thesis, and not to provide an exhaustive description over the wide variety of metaheuristics algorithms, we will just focus on the simulated annealing algorithm.

Simulated annealing (SA) is a metaheuristic for the combinatorial optimization problem. It can easily explore a big search space and it is not affected by bad candidate solutions as these are simply discarded. The algorithm does not need any information about the problem, since it only requires to know how to com-

pare to different solutions. The SA algorithm has been proposed by Kirkpatrick et al. [KGV83] and it is an evolution of the algorithm proposed by Metropolis et al. [MRR+53].

The algorithm proposed by Metropolis et al. [MRR+53] is based on the concept of iterative improvement. Each iteration introduces small changes to the current solution in order to (slightly) reduce the value of an objective function (energy). At each step the algorithm checks if the solution is improved. If this is the case, then the new solution replaces the previous one as starting point. Otherwise the new solution is accepted with a probability proportional to the difference of energy between the current solution and the previous solution. The algorithm terminate when a good solution is found or after a given number of iterations.

Kirkpatrick et al. [KGV83] extended the algorithm of Metropolis et al. with the concept of cooling down typical of annealing in metallurgy. The probability of accepting a solution is now also related to a temperature. The temperature initially is high and decreases with the number of iterations. The introduction of temperature allows for a time bound where later solutions are more likely to be better than earlier ones. Finally, Geman and Geman [GG84] proved that the SA algorithm converges to the global optimum if annealed sufficiently slowly.

The original version of the SA algorithm has been defined for single-objective optimization problems. In our context we will deal with multi-objective optimization problems since is plausible to believe that more than one risk will be defined for a process model. In the case of multi-objective optimization problems the original simulated annealing algorithm cannot be used. Several solutions [UTO98, CJ98, SSPC00, Sum03, SEF+08] have been proposed which overcome this limitation.

Smith et al. [SEF+08] introduce the concept of dominance and non dominance between two solutions. Given two solutions a and b, a dominates b, i.e. $a \prec b$, if there is at least a value of an objective function of a that is lower than the corresponding value in b, and all the other objective functions have the same values (or lower). As consequence if neither solution dominates the other, they are non-dominating, i.e. $a \not\prec b$ and $b \not\prec a$. Using these two concepts, Smith et al. propose an extension of the simulated annealing algorithm that uses the relationship between dominating and mutually non dominating solutions. In their approach the difference of energy is calculated taking into account the number of solutions that the elegible solution dominates. Finally, as pointed out by Suman [Sum04] the algorithm proposed by Smith et al. requires less computational cost than other approaches [UTO98, CJ98, SSPC00, Sum03, SEF+08], and generates well

diversified solutions.

3.4 Summary

In this chapter we introduced a number of important concepts and their definitions used in the following chapters. In particular, we presented the concept of YAWL specification and its formal representation. A YAWL specification is a set of YAWL nets that together model a business process including activities, resources, and data that are utilized during the execution of the process. This concept will allow us have a better understanding of the techniques that will be proposed in the following chapters, since all techniques are implemented on top of (but not limited to) the YAWL system.

We then introduced the concepts of event log and YAWL log. An event log is the set of events that are generated during the execution of an automated business process and are store during the logging process. In its minimal form, an event should contain the name activity and the id of the process instance generating the event. Additional information can be provided such as in the case of the YAWL log, where also information about resources, timestamps, and data are provided.

Finally, we provided a briefly introduction to operation research techniques with an emphasis on linear programming and simulated annealing. Both techniques are used to solve optimization problems. In particular, linear programming is a mathematical approach for the solution of optimization problems where both objective function and constraints are linear. Simulated annealing on the other hand is a metaheuristic approach used to optimization problems via a probabilistic search algorithm.

In the next chapter we present an architecture for risk monitoring based on sensors, which is the first component of our risk-aware framework.

Chapter 4

Risk Definition and Monitoring

In the Introduction we highlighted that effective risk management is imperative for organizational survival, and incidents such as those described in the Introduction reveal that proper integration of risk management with process management is now vital. Further, a focus on risk analysis alone is no longer adequate. Rather, active, real-time risk detection, analysis and mitigation is required. Real-time risk detection is particularly essential in mission-critical business processes or those involving life-saving activities [CMZ09, MZW12].

In this chapter we present a technique for real-time monitoring of risks in executable business process models, as part of the risk-aware approach described in Section 1.4. This is achieved by incorporating risk management into all phases of the BPM lifecycle: from process design, where high-level risks defined via a risk analysis method are mapped down to specific process model elements such as activities, resources and data, through to process diagnosis, where risks are detected during process execution. By automating risk detection, the interested users (e.g. a process administrator or a risk manager) can be notified as early as a risk is detected (i.e. in real-time), such that remedial actions can be taken to rectify the current process instance, and prevent an undesired state of the process (*fault* for short), from occurring. Based on historical data, we can also compute the probability of a risk at run-time, and compare it to a dynamic threshold, so as to notify the user about the severity of a risk, when the latter is no longer tolerable. The dynamic threshold is computed on the fly, allowing the definition of different threshold values based for example on the resource performing a specific task or on the value of a specific variable (e.g. the threshold is 70% if the premium of a claim is below $1000 or 50% otherwise). This threshold could also be dynamically set as the *upper bound likelihood* for a given process risk. This upper bound value can be computed as the average risk likelihood plus its standard

deviation, measured over a time window (e.g. within one year) from the logs, and updated regularly based on the needs of the company under exam. In this case the technique will identify risky process behaviour as the process instances with a risk likelihood greater than this upper bound, i.e. the outliers. Moreover, trends observed in the behaviour of a process (i.e. an increase or decrease of the value of the average risk likelihood over time), can serve as indicators to detect changes in the risk level of a process (riskier for ascending trends or safer for descending trends), and as such, they could be used to adapt the dynamic threshold accordingly.

Our technique is operationalized via a distributed, sensor-based architecture on top of a BPMS. Each sensor is coupled with a risk condition capturing the situation upon which the risk of a given fault may occur. Sensors are defined at design-time on the executable process model. Conditions can be determined via a query language that can fetch both historical and current execution data from the logs of the BPMS. At run-time sensors are registered with a central sensor manager. At a given sampling rate, or based on the occurrence of a specific event, the sensor manager retrieves and filters all data relevant for the various sensors (as it is logged by the BPMS engine), and distributes it to the relevant sensors. If a sensor condition holds, i.e. if the probability of the associated risk is above a given threshold, the sensor alerts the sensor manager which in turn notifies the monitoring component of the BPMS. The distributed nature of the architecture guarantees that there is no performance overhead on the BPMS engine, and thus on the execution of the various process instances. We implemented this architecture on top of the YAWL system [HAAR10]. We extended the YAWL Editor to cater for the design of risk sensors, and equipped the run-time environment with a sensor manager service that interacts with YAWL's monitoring service and execution engine. Finally, to facilitate the definition of process risks, we implemented a set of risk templates for various categories, such as organizational, data-related and technology-related. Such templates are abstract risk definitions which users need to bind to concrete process elements.

To prove the feasibility of the proposed approach, we used fault tree analysis [Com90] (a well-established risk analysis method) to identify risk conditions in a reference process model for logistics, in collaboration with an Australian risk consultant. These risks embrace different process aspects such as tasks' order dependencies, involved resources and business data, and relate to historical data where needed, to compute risk probabilities. We expressed these conditions via sensors in the YAWL system, and measured the time needed to compute these

conditions at run-time. The tests showed that the sensor conditions can be computed in a matter of milliseconds without impacting on the performance of the running process instances. Finally, we conducted a usability analysis of the approach and its implementation, by inviting 21 users to report on their experiences while using the approach for analyzing and defining risk sensor conditions. The results from this experiment show that the approach is being perceived as interesting and useful, especially by novice users, confirming an intention of the participants to use it.

This chapter is organized as follows. Section 4.1 illustrates the running example in the logistics domain. Section 4.2 presents the sensor-based architecture to implement this approach. Section 4.3 formalizes the abstract syntax of the language proposed for risk detection while Section 4.4 shows how the risks defined on the running example can be modeled via this language. Next, Section 4.5 proposes a set of risk templates to facilitate the definition of process risks. Section 4.6 illustrates the implementation of the sensor-based architecture, while Section 4.7 reports on the results of the performance and usability evaluations. Section 4.8 covers related work while Section 4.9 concludes the chapter.

The research presented in this chapter has been the subject of several publications [CFLH11, CLF+13b, CLF13a, CLH+13].

4.1 Running Example

In this section we use an example to illustrate how the risk of possible faults to occur during a business process execution can be detected as early as possible. In particular, we show how risks can be expressed in terms of process-specific aspects such as tasks occurrence, data or available resources. Figure 6.1 describes the payment subprocess of an order fulfillment business process which is inspired by the VICS industry standard for logistics [Vol11]. The notation used to represent this example is that of YAWL (see Chapter 3), although a deep knowledge of this language is not required.

This process starts after the freight has been picked up by a carrier and deals with the shipment payment. The first task is the production of a Shipment Invoice containing the shipment costs related to a specific order for a specific customer. If shipments have been paid in advance, all that is required is for a Finance Officer to issue a Shipment Remittance Advice specifying the amount being debited to the customer. Otherwise, the Finance Officer issues a Shipment Payment Order that needs to be approved by a Senior Finance Officer (who is

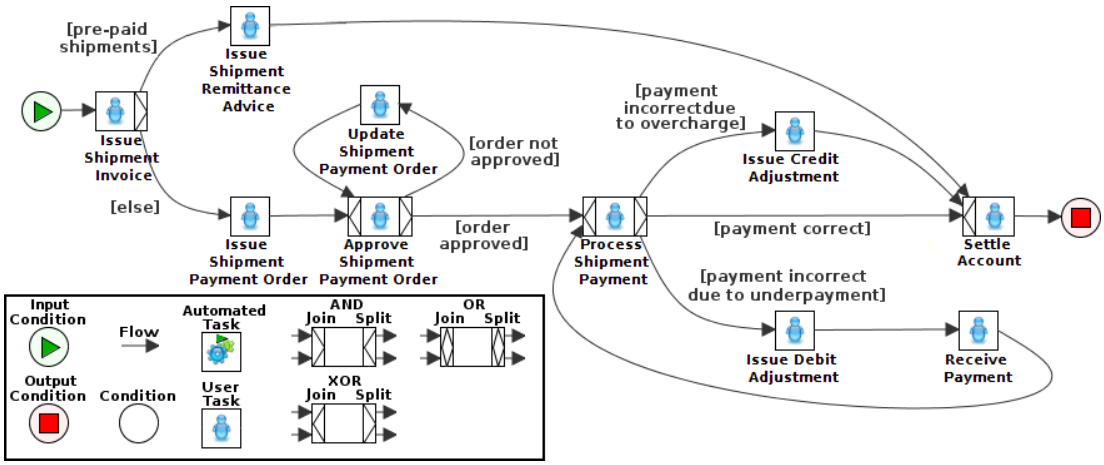


Figure 4.1: Order-Fulfillment: Payment subprocess.

the superior of this Finance Officer). At this point, a number of updates may be made to the Shipment Payment Order by the Finance Officer that issued it, but each of these needs to be approved by the Senior Finance Officer. After the document is finalized and the customer has paid, an Account Manager can process the shipment payment by specifying the balance. If the customer underpaid, the Account Manager needs to issue a Debit Adjustment, the customer needs to pay the balance and the payment needs to be reprocessed. A customer may also overpay. In this case the Account Manager needs to issue a Credit Adjustment. In the latter case and in case of a correct payment, the shipment payment process is completed.

In collaboration with a risk analyst of an Australian consulting company, we identified four faults that can occur during the execution of this payment subprocess. In order to prevent the occurrence of these faults, for each of them we also defined an associated risk condition by using fault tree analysis [Com90]. Accordingly, each risk condition is expressed as a set of lower-level boolean events which are organized in a tree via logical connectives such as ORs, ANDs and XORs, an example of fault tree is shown in Figure 4.2.

The first fault is an *overtime process* fault. A Service Level Agreement (SLA) for a process or for a given task within a process, may establish that the process (or task) may not last longer than a Maximum Cycle Time MCT, otherwise the organization running the process may incur a pecuniary penalty. In our case, an overtime fault occurs if an instance of the payment subprocess is not completed within an MCT of five days.

To detect the risk of overtime fault at run-time, we should check the likelihood that the running instance does not exceed the MCT based on the amount of

time T_c expired at the current stage of the execution. Let us consider T_e as the remaining cycle time, i.e. the amount of time estimated to complete the current instance given T_c, then the probability of exceeding MCT can be computed as $1 - (MCT - T_c)/T_e$ if $T_c < MCT < T_e + T_c$, it is 1 if $T_c \geq MCT$, and it is 0 if $T_e + T_c \leq MCT$.[1] If this probability is greater than a tolerance value (e.g. 60%), we notify the risk to the user. The estimation of the remaining cycle time is based on past executions of the same process and can be computed using the approach of Van der Aalst et al. [ASS11], whereby using an annotated transition system an estimation of the remaining cycle is calculated by replaying the trace on the transition system and returning the expected time annotated on the last node visited (see Section 4.7 for more details).

The second fault is related to the resources participating in the process. The Senior Finance Officer who has approved a Shipment Payment Order for a given customer, must have not approved another order by the same customer in the last d days, otherwise there is an *approval fraud*. This fault is thus generated by the violation of a four-eye principle across different instances of the Payment subprocess.

To detect the risk of this fault we first have to check that there is an order, say order o of customer c, to be approved. This means checking that an instance of task Approve Shipment Payment Order is being executed. Moreover, we need to check that either of the following conditions holds: i) o has been allocated to a Senior Finance Officer who has already approved another order for the same customer in the last d days; or ii) at least one Senior Finance Officer is available who approved an order for customer c in the last d days and all other Senior Finance Officers who never approved an order for c during the last d days are busy. The corresponding fault tree is shown in Figure 4.2.

The third fault relates to a situation where a process instance executes a given task too many times. This situation typically occurs in the context of loops. Not only could this lead to a process slowdown but also to a "livelock" if the task is in a loop whose exit condition is purposefully never met. In general, given a task t a maximum number of allowable executions of t per process instance $MAE^i(t)$ can be fixed as part of the SLA for t. With reference to the Payment subprocess, this can occur for example if task Update Shipment Payment Order is re-executed five times within the same process instance. We call this an *order unfulfillment* fault.

[1] A more accurate value could be obtained considering various scenarios or paths in the transition system [ASS11] to see what fraction of these paths will lead to exceeding the MCT.

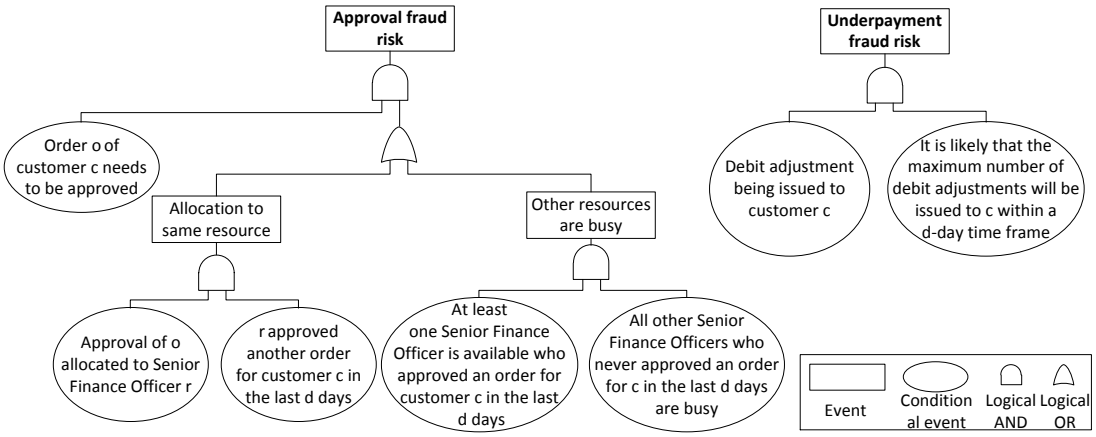


Figure 4.2: The fault trees for Approval Fraud and Underpayment Fraud.

To detect the risk of this fault at run-time, we need to check if: i) an order o is been updated (i.e. task Update Shipment Payment Order is currently being performed for order o); and ii) it is likely that this order will be updated again (i.e. task Update Shipment Payment Order will be repeated within the same process instance). The probability that the number of times a task will be repeated within the same instance of the Payment subprocess is computed by dividing the number of instances where the MAE^i for task Update Shipment Payment Order has been reached, over the number of instances that have executed this task at least as many times as it has been executed by the current instance, and have completed. The tolerance value indicates a threshold above which the risk should be notified to the user. For example, if this threshold is 60% for task t, a risk should be raised if the probability of $MAE^i(t)$ is greater than 0.6.

The fourth fault is an *underpayment fraud*. It relates to a situation in which a given task is executed too many times across multiple process instances. Similar to the previous fault, given a task t we can define a maximum number of allowable executions of t per process $MAE^p(t)$ as part of the SLA for p. In our example, this type of fault occurs when a customer underpays more than three times within the last five days.

To detect the risk of underpayment fraud, we need to check if: i) a debit adjustment is currently being issued to a customer c (i.e. task Issue Debit Adjustment is currently being performed for customer c); and ii) it is likely that the maximum number of debit adjustments will be issued to the same customer in a d-day time frame. The probability that MAE^p is reached for task Issue Debit Adjustment of customer c in d days is computed by dividing the number of customers for which the MAE^p for task Issue Debit Adjustment has been reached within d days, over the number of customers for which this task has been ex-

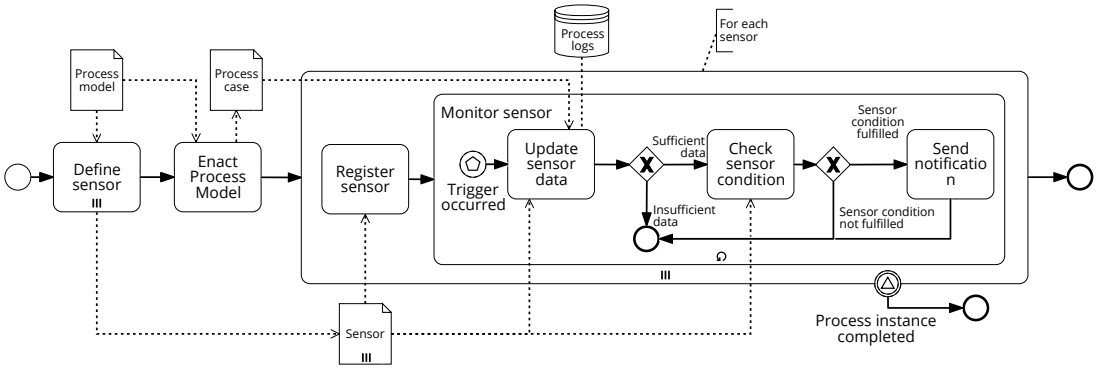


Figure 4.3: Realization of risk-aware BPM lifecycle via sensors.

ecuted at least as many times as it has been executed for c within d days. If this probability is above a tolerance value, the risk should be raised and the user notified. Similar to the previous risk, the tolerance value indicates a threshold above which this risk should be notified to the user. The corresponding fault-tree is shown in Figure 4.2.

Finally, these four risks will clearly have a different impact on the process and the organization executing the process. In order to differentiate the consequence that each risk may have on the organization we can define a *dynamic consequence* for each one of them. For example, for the overtime process the monetary consequence could be proportional to the amount of time exceeding the SLA, plus an initial fee.

In the next section we describe a sensor-based architecture to operationalize our technique, according to the enhanced BPM lifecycle described in Section 1.4.

4.2 Sensor-based Realization

In order to realize our risk-aware BPM lifecycle, we devised a technique based on sensors. In a nutshell, the idea is to capture risk and fault conditions via sensors, and then monitor these sensors during process execution. An overview of this approach is shown in Figure 4.3 using the BPMN 2.0 notation [Obj11].

Sensors are defined during the *Process Design* and *Process Implementation* phases of our risk-aware BPM lifecycle (see Figure 1.3), for each process model for which the presence of risks and/or faults need to be monitored. If the process model is specified via an executable language, then these two phases coincide.

A sensor is defined through a boolean *sensor condition*, constructed on a set of process variables, and a *sensor activation trigger*. Process variables are used to retrieve information from the specific instance in which the sensor condition will

be evaluated as well as from other instances, either completed or still running. For example, we can use variables to retrieve the resource allocated to a given task, the value of a task variable, or the status of a task. Process instances can either be identified based on the current instance (e.g. the last five instances that have been completed before the current one), or based on the fulfillment of a *case condition* (e.g. "all instances where a given resource has executed a given task"). The sensor condition can represent either a *risk condition* associated with a fault, or a *fault condition*, or both. If both conditions are specified, the fault condition is evaluated only if the risk condition evaluates to true. For example, the sensor will check if an overtime process fault has occurred in a process instance only if first the risk of such fault has first been detected, based on the estimation of the remaining cycle time for this instance. Finally, the sensor activation trigger can be either a timer periodically fired according to a sampling rate (e.g. every 5 minutes), or an event emitted by the process engine (e.g. the completion of a task).

During *Process Enactment*, the defined sensors are registered with a *sensor manager*, which activates them. In the *Process Diagnosis* phase, which starts as soon as the process is enacted, the activated sensors receive updates on the variables of their sensor conditions according to their trigger (timer or event). When a sensor receives an update, it checks its sensor condition. If the condition holds, a notification is sent from the sensor to the monitor service of the BPMS, informing the process administrator about the eventuation of a risk and its consequence (e.g., a monetary consequence).

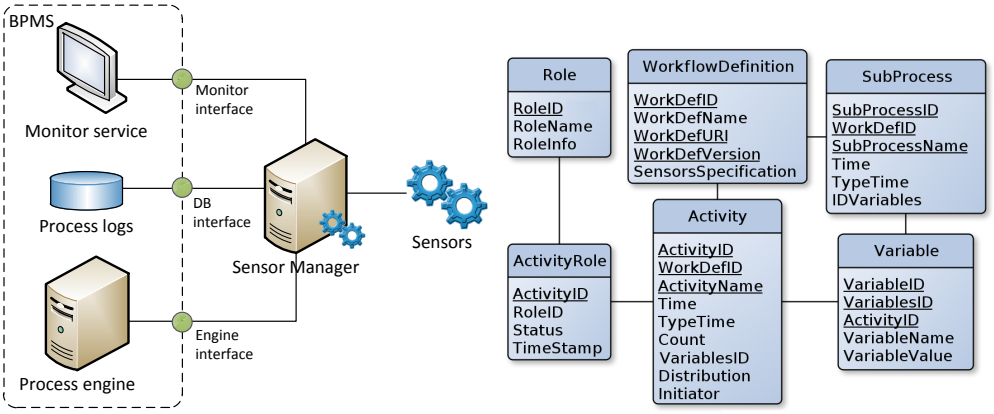
The sensor manager relies on three interfaces to interact with the BPMS (see Figure 4.4a):

- *Engine interface*, used to register a sensor with a particular event raised by the BPMS engine. When the event occurs the sensor is notified by the sensor manager.

- *Database interface*, used to query the BPMS database in order to collect current and historical information.

- *Monitor interface*, used to notify the detection of risks and faults to the monitor service of the BPMS.

These interfaces can be implemented by the vendor or user of the BPMS where the sensor manager needs to be installed. In this way, our sensor manager can virtually be interfaced with any BPMS. As an example, the conceptual model of



(a) Sensor-based architecture. (b) Database Interface schema model.

the database interface is showed in Figure 4.4b, where methods have been omitted for space reasons. This conceptual model is inspired by the reference process meta-model of the WfMC (see Section 2.1), in order to cover as many aspects as possible of a workflow model, and meantime, to remain as generic as possible. For example, class *WorkFlowDefinition* allows one to retrieve information about the process model where the sensor is defined, such as process identifier and name, while class *SubProcess* allows one to retrieve information about a specific subprocess, and so on. This interface should be implemented according to the characteristics of the specific database used in the BPMS at hand. For an efficient use of the interface, one should also define indexes on the attributes of the BPMS database that map the underlined attributes in Figure 4.4b. These indexes have been determined based on the types of queries that can be defined in our sensor definition language.

An alternative approach to achieve the portability of the sensor manager, would be to read the BPMS logs from a standard serialization format such as OpenXES [GV13]. However, as we will show in Section 4.7, this solution is rather inefficient.

The advantages of using sensors are twofold. First, their conditions can be monitored while the process model is being executed, i.e. in real-time. Second, according to a distributed architecture, each sensor takes care of checking its own condition after being activated by the sensor manager. In this way, potential execution slowdowns are avoided (e.g., the process engine and the sensor manager could be deployed onto two different machines).

4.3 Sensor Definition Language: Abstract Syntax

In this section we describe the abstract syntax [Mey90] of the sensor definition language. A detailed description of all elements of this abstract syntax is reported in Appendix A. The sensor definition language shares common aspects with programming languages such as with query languages, since it is inspired to JAVA from a syntactic point of view and to SQL for its semantics. During the definition of the sensor definition language we took in consideration the design principles discussed in [HB99, Ben86], in particular: i) expressibility: is the language expressive?; ii) completeness: can the language describe all objects of interest?; iii) extensibility: can the language be easily extended?; iv) formal foundation: is the language formally defined in order to guarantee unambiguity and executability?.

The definition of a sensor requires a *risk condition*, a boolean *fault condition*, a *sensor activation trigger*, and a *consequence* which represents the gravity of the impact the fault will have on the company in case it occurs (e.g. in monetary terms). The trigger can be a timer based on a sampling rate, or an event produced by the engine interface. Risk condition and fault condition are constructed on a set of process variables. A variable is defined with a mapping among a *varName* and an *Info*.

$$
\begin{aligned}
\mathit{Sensor} &\triangleq v : \mathit{Variables}; t : \mathit{Trigger}; \mathit{riskCond} : \mathit{RiskCon};\\
&\quad\ \mathit{faultCond} : \mathit{BoolCon}; \mathit{consequence} : \mathit{MaCon};\\
\mathit{Trigger} &\triangleq \mathit{timer} \mid \mathit{event}\\
\mathit{Variables} &\triangleq \mathit{Assignment}^{+}\\
\mathit{Assignment} &\triangleq \mathit{result} : \mathit{varName}; i : \mathit{Info}
\end{aligned}
$$

This *Info* can be a constant (using *Definition*), or the result of a function executed on a variable (using *VarFun*), or a piece of information collected from the process instance (using *CaseExp* or *CaseEleExp*). We use a *CaseExp* if the information is related to the process instance itself, while a *CaseEleExp* if the information is related to an element of a process instance (that must be specified using *TaskOrNet* that identifies a task by taskLabel or a net by netName).

$$
\begin{aligned}
\mathit{Info} &\triangleq \mathit{Definition} \mid \mathit{VarFun} \mid \mathit{CaseExp} \mid \mathit{CaseEleExp} \\
\mathit{VarFun} &\triangleq \mathit{ResCon} \mid \mathit{ResSimFun} \mid \mathit{ResComFun} \\
\mathit{Definition} &\triangleq c : \mathit{constant} \\
\mathit{CaseEleExp} &\triangleq \mathit{ce} : \mathit{CaseExp}; \mathit{ton} : \mathit{TaskOrNet} \\
\mathit{TaskOrNet} &\triangleq \mathit{taskLabel} \mid \mathit{netName}
\end{aligned}
$$

When we use a *CaseExp* we must specify the instances of interest (using *CaseIDStat*), and the action that identifies the piece of information (using *Action*). Such *Action* can either be an information related to a predicate function, a predicate function with input, a task or net variable (i.e. a variable at the level of tasks or a variable at the level of processes), or a task or net subvariable (i.e. a subvariable of a task variable or a net variable, respectively). An instance can be identified in various ways, by its position among all the instances of the same process model (using *absExp*), by its position respect the current instance (using *relExp*), or by the fulfillment of some conditions (using *CaseConSet*).

$$
\begin{aligned}
\mathit{CaseExp} &\triangleq \mathit{cis} : \mathit{CaseIDStat}; a : \mathit{Action} \\
\mathit{CaseIDStat} &\triangleq \mathit{absExp} \mid \mathit{relExp} \mid \mathit{CaseConSet} \\
\mathit{Action} &\triangleq \mathit{predAct} \mid \mathit{inputPredAct} \mid \mathit{taskOrNetVar} \mid \mathit{SubVarExp} \\
\mathit{SubVarExp} &\triangleq \mathit{var}^{+}
\end{aligned}
$$

These conditions can be on the identifier of the process instance (through the use of *CaseParam*), or on a piece of information related to a task or a net (using *CaseCon*). It is also possible to specify multiple conditions that are obtained by the conjunction of several *CaseParam* and *CaseCon* elements (using *CaseConExp*).

$$
\begin{aligned}
\mathit{CaseConSet} &\triangleq \mathit{CaseCon} \mid \mathit{CaseParam} \mid \mathit{CaseConExp} \\
\mathit{CaseConExp} &\triangleq \mathit{ccs}_1, \mathit{ccs}_2 : \mathit{CaseConSet}; \mathit{bo} : \mathit{BoolOp} \\
\mathit{CaseCon} &\triangleq \mathit{ton} : \mathit{TaskOrNet}; a : \mathit{Action}; \mathit{co} : \mathit{CompOp}; \mathit{rhe} : \mathit{RHExp} \\
\mathit{CaseParam} &\triangleq i : \mathit{idFun}; \mathit{co} : \mathit{CompOp}; \mathit{rhe} : \mathit{RHExp} \\
\mathit{CompOp} &\triangleq \mathit{le} \mid \mathit{leq} \mid \mathit{ge} \mid \mathit{geq} \mid \mathit{eq} \mid \mathit{contains} \mid \mathit{isContained}
\end{aligned}
$$

Whether using a *CaseParam* or a *CaseCon* it will be compared with a *RHExp*. A *RHExp* can be a *constant*, a *function*, a *varName*, or an expression containing

those elements (using *RHExpSet*).

$$\begin{aligned} RHExp &\triangleq constant \mid function \mid varName \mid RHExpSet \\ RHExpSet &\triangleq rhe_1, rhe_2 : RHExp; o : Op \end{aligned}$$

Once all the variables have been specified the risk condition can be defined. A *RiskCon* is composed of two *MaCon* elements, one is the risk likelihood and the other is the risk threshold. A risk condition evaluates true if the likelihood exceeds the threshold. A *MaCon* is an arithmetical expression that can use constants, variables, results of functions invoked on variables (using *ResSimFun*, and *ResComFun*), and the results obtained by the execution of loops (using *MaFor*, and *FixMaFor*). A *MaCon* may be in a conditional form, represented as *MaITE*, or be a normal expression represented as *MaExp*, despite it is always possible to have conditional elements inside a normal expression. A conditional expression *MaITE* is composed of a *BoolCon* representing the *if* and two *MaCon*s representing the *then* and *else* clauses, respectively.

$$\begin{aligned} RiskCon &\triangleq riskL, riskT : MaCon \\ MaCon &\triangleq MaITE \mid MaFor \mid ResSimFun \mid MaExp \mid \\ &\quad FixMaFor \mid constant \mid ResComFun \mid varName \\ MaITE &\triangleq if : BoolCon; then, else : MaCon \end{aligned}$$

From the definition of the *MaExp* element on, the syntax describes an arithmetical expression. In fact a *MaExp* element can be solved via the analysis of an unary expression *MaUnExp* (i.e. a negative expression) or a binary expression *MaBinExp*, that is an arithmetical operation between two *MaCon*s.

$$\begin{aligned} MaExp &\triangleq MaUnExp \mid MaBinExp \\ MaUnExp &\triangleq s : sub; me : MaCon \\ MaBinExp &\triangleq me_1, me_2 : MaCon; mo : MaOp \\ MaOp &\triangleq add \mid sub \mid mul \mid div \mid exp \mid mod \end{aligned}$$

As said before a condition expression is defined using a *BoolCon*. A *BoolCon* is a boolean expression that can use constants, variables, results of functions invoked on variables (using *ResCon*), and the results obtained by the execution of loops (using *BoolFor*, and *FixBoolFor*). A *BoolCon* may be in a conditional form,

represented as *BoolITE*, or be a normal expression represented as *BoolExp*, despite it is always possible to have conditional elements inside a normal expression. A conditional expression *BoolITE* is composed of three *BoolCon*s representing the *if*, the *then*, and the *else*.

$$
\begin{aligned}
\mathit{BoolCon} &\triangleq \mathit{BoolITE} \mid \mathit{BoolExp} \mid \mathit{BoolFor} \mid \mathit{FixBoolFor} \mid \\
&\quad\ \mathit{Comp} \mid \mathit{ResCon} \mid \mathit{varName} \mid \mathit{constant} \\
\mathit{BoolITE} &\triangleq \mathit{if}, \mathit{then}, \mathit{else} : \mathit{BoolCon}
\end{aligned}
$$

The *BoolExp* element represents a boolean expression that is describes from this point on. In fact a *BoolExp* element can be solved via the analysis of an unary expression *BoolUnExp* or a binary expression *BoolBinExp*, that can contain the result of a comparison *Comp*, and so on.

$$
\begin{aligned}
\mathit{BoolExp} &\triangleq \mathit{BoolUnExp} \mid \mathit{BoolBinExp} \\
\mathit{BoolUnExp} &\triangleq n : \mathit{neg}; e : \mathit{BoolCon} \\
\mathit{BoolBinExp} &\triangleq e_1, e_2 : \mathit{BoolCon}; \mathit{bo} : \mathit{BoolOp} \\
\mathit{BoolOp} &\triangleq \mathit{and} \mid \mathit{or} \\
\mathit{Comp} &\triangleq \mathit{ce}_1, \mathit{ce}_2 : \mathit{CompElem}; \mathit{co} : \mathit{CompOp} \\
\mathit{CompElem} &\triangleq \mathit{MaCon} \mid \mathit{ResListFun} \mid \mathit{varName} \mid \mathit{constant} \\
\mathit{CompOp} &\triangleq l \mid \mathit{leq} \mid \mathit{eq} \mid \mathit{geq} \mid g \mid \mathit{noteq}
\end{aligned}
$$

The functions that can be invoked on a variable are identified by the elements: *ResCon*, *ResListFun*, *ResSimFun*, and *ResComFun*. These elements can be used only in specific points of the syntax since the result returned can be a boolean, or a list, or a number.

$$
\begin{aligned}
\mathit{ResCon} &\triangleq \mathit{res} : \mathit{varName}; a : \mathit{Activity} \\
\mathit{ResListFun} &\triangleq \mathit{result} : \mathit{varName}; \mathit{lrf} : \mathit{listResFun} \\
\mathit{ResSimFun} &\triangleq \mathit{resource} : \mathit{varName}; \mathit{srf} : \mathit{simResFun} \\
\mathit{ResComFun} &\triangleq \mathit{res}_1, \mathit{res}_2 : \mathit{varName}; \mathit{crf} : \mathit{comResFun} \\
\mathit{Activity} &\triangleq \mathit{resource} : \mathit{varName}; \mathit{lrf} : \mathit{ResListFun}
\end{aligned}
$$

The constructs *MaFor* and *BoolFor* are used to execute nested loops. The definition of one of these elements requires to specify the type of loop (*TypeBF* for *BoolFor* or *TypeMF* for *MaFor*), the list of variables (i.e. *ListResult*) that

will be used to create the nested loops (one loops for each variable), and the expression that must be executed. The constructs *FixMaFor* and *FixBoolFor* are nested loops like *MaFor* and *BoolFor* for which the variable *fixEle* determines the number of loops that need to be executed. The *TypeBF* and *TypeMF* describe how the results of each execution will be joined. It is possible to choose among four types of loops: and (i.e. logic AND), or (i.e. logic OR), add (i.e. addition), and mul (i.e. multiplication).

$$
\begin{aligned}
\mathit{MaFor} &\triangleq t : \mathit{TypeMF}; lr : \mathit{ListResult}; \mathit{fme} : \mathit{ForMaCon} \\
\mathit{FixMaFor} &\triangleq \mathit{fixEle} : \mathit{varName}; mf : \mathit{MaFor} \\
\mathit{TypeMF} &\triangleq \mathit{add} \mid \mathit{mul} \\
\mathit{BoolFor} &\triangleq t : \mathit{TypeBF}; lr : \mathit{ListResult}; \mathit{fbe} : \mathit{ForBoolCon} \\
\mathit{FixBoolFor} &\triangleq \mathit{fixEle} : \mathit{varName}; bf : \mathit{BoolFor} \\
\mathit{TypeBF} &\triangleq \mathit{and} \mid \mathit{or} \\
\mathit{ListResult} &\triangleq \mathit{varName}^{+}
\end{aligned}
$$

As for a condition also for loops it is possible to assume conditional or normal forms. The expression executed inside a loop is similar to the condition expression but with some variations. The *ForMaCon* is a arithmetical expression that can use constants and variables, the difference with an *MaCon* is that it is not possible to use loops or functions inside this type of expression.

$$
\begin{aligned}
\mathit{ForMaCon} &\triangleq \mathit{ForMaITE} \mid \mathit{ForMaExp} \mid \mathit{constant} \mid \mathit{varName} \\
\mathit{ForMaITE} &\triangleq \mathit{if} : \mathit{ForBoolCon}; \mathit{then}, \mathit{else} : \mathit{ForMaCon} \\
\mathit{ForMaExp} &\triangleq \mathit{ForMaUnExp} \mid \mathit{ForMaBinExp} \\
\mathit{ForMaUnExp} &\triangleq s : \mathit{sub}; \mathit{me} : \mathit{ForMaCon} \\
\mathit{ForMaBinExp} &\triangleq \mathit{me}_1, \mathit{me}_2 : \mathit{ForMaCon}; \mathit{mo} : \mathit{MaOp}
\end{aligned}
$$

The *ForBoolCon* is a boolean expression and does not allow the use of loops or functions as the *ForMaExp*. Clarified this two points, all the elements the name of which starts with *For* are equals to the elements used by an *BoolCon*.

$$
\begin{array}{rcl}
\mathit{ForBoolCon} & \triangleq & \mathit{ForBoolITE} \mid \mathit{ForBoolExp} \mid \mathit{ForComp} \mid \\
 & & \mathit{varName} \mid \mathit{varName} \mid \mathit{constant} \\
\mathit{ForBoolITE} & \triangleq & \mathit{if}, \mathit{then}, \mathit{else} : \mathit{ForBoolExp} \\
\mathit{ForBoolExp} & \triangleq & \mathit{ForBoolUnExp} \mid \mathit{ForBoolBinExp} \\
\mathit{ForBoolUnExp} & \triangleq & n : \mathit{neg}; e : \mathit{ForBoolCon} \\
\mathit{ForBoolBinExp} & \triangleq & e_1, e_2 : \mathit{ForBoolCon}; \mathit{bo} : \mathit{BoolOp} \\
\mathit{ForComp} & \triangleq & \mathit{ce}_1, \mathit{ce}_2 : \mathit{ForCompElem}; \mathit{co} : \mathit{CompOp} \\
\mathit{ForCompElem} & \triangleq & \mathit{ForMaCon} \mid \mathit{varName} \mid \mathit{constant}
\end{array}
$$

4.4 Risk Definition for the Running Example

The abstract syntax defined in Section 4.3 has been used as a formal foundation for the definition of the concrete syntax of our language for sensor conditions. During the realization of the concrete syntax we followed the dot notation typical of object-oriented languages, whereby a "." is used as a separator between the object on which a function is requested and the function itself. In particular, whenever a function "a()" needs to be invoked on an object "b", the structure used is "b.a()". In the context of our language we follow this hierarchy: "case.task.action()", "case.net.action()", or "case.action()" for the definition of variables, and "task.action()", or "net.action()" for the identification of cases which satisfy certain conditions. Finally, another aspect in common with object-oriented languages is the declaration of variables. In our concrete syntax, conditions can only be defined using variables that have been previously declared and defined using a mapping.

Using this concrete syntax, we now have all ingredients to show how the risks that we identified for the Payment subprocess can be captured via sensor conditions. There is an overtime process if an instance of the payment subprocess is not completed within an MCT of five days (see Section 4.1). This condition is checked by using two variables: one to retrieve the amount of time T_c expired and the other to retrieve the T_e remaining cycle time.

$$
\begin{array}{rcl}
d & = & 5 \\
T_c & = & \mathsf{case(current).Payment(PassTimeInMillis)} \\
T_e & = & \mathsf{case(current).(TimeEstimationInMillis)}
\end{array}
$$

Assuming a tolerance value of 60%, the risk condition defined to monitor this risk is:

$$(T_c \geq (d * 24 * 60 * 60000)) \vee ((d * 24 * 60 * 60000) < (T_e + T_c) \wedge (1 - ((d * 24 * 60 * 60000) - T_c)/T_e) > 0.6)). \quad (4.1)$$

There is an approval fraud whenever a Senior Finance Officer (SFO) approves two orders for the same customer within five days (see Section 4.1). Accordingly, the corresponding risk can be detected if given an order o of customer c to be approved, i) o is allocated to a SFO who approved another order for the same customer in the last five days; or ii) the only SFOs available are the ones who approved an order for customer c in the last five days.

This risk condition is triggered by an event, i.e. the spawning of a new instance of task Approve Shipment Payment Order (see Figure 6.1). This is checked using a variable to retrieve the status of this task in the current instance. The risk condition itself is given by the disjunction of the two conditions described above. The first condition is checked by using variable *ASPO#* to retrieve the number of times this task was completed for customer c by the resources to whom is currently allocated. Variable *ASPO#* is defined via a case condition over customer c, the resource to whom is currently allocate (variable sfo_1), the completion time of this task (that must be greater than the start time of the current task Approve Shipment Payment Order minus five days in milliseconds[2]), and the identifier of the instance (that must be different from the identifier of the current instance).

The second condition is checked by using two variables and invoking two functions. Variable sfo_2 retrieves which resources completed task Approve Shipment Payment Order, and variable *sfo* retrieves all resources that can be offered this task (i.e. the current task). The first variable is defined via a case condition over customer c and the completion time of this task (that must be greater than the start time of the current task Approve Shipment Payment Order minus five days). The two invoked functions return the number of tasks started on the resources that completed task Approve Shipment Payment Order, and the number of tasks in the execution queue of the resources that have been offered this task, and did not complete it for customer c in the last five days.

After the definition of the variables, defined in Table 4.1, the risk condition is

[2]Time is measured in milliseconds as logging systems save timestamps in this unit, and this facilitates the comparison of timings.

sfo_1	=	case(current).Approve_Shipment_Payment_Order_593(allocateResource)
c	=	case(current).Issue_Shipment_Invoice_594.ShipmentInvoice.Company
d	=	5
$ASPO$	=	case(current).Approve_Shipment_Payment_Order_593(OfferTimeInMillis)
$ASPO_{ST}$	=	case(current).Approve_Shipment_Payment_Order_593(StartTimeInMillis)
$ASPO\#$	=	case(Approve_Shipment_Payment_Order_593(completeResource)=sfo_1 ∧ Issue_Shipment_Invoice_594.ShipmentInvoice.Company=c ∧ Approve_Shipment_Payment_Order_593(CompleteTimeInMillis)> ($ASPO$-(d*24*60*60*1000)) ∧ (ID)!=[IDCurr]) .Approve_Shipment_Payment_Order_593(CountElements)
sfo_2	=	case(Issue_Shipment_Invoice_594.ShipmentInvoice.Company=c ∧ Approve_Shipment_Payment_Order_593(isCompleted)="true" ∧ Approve_Shipment_Payment_Order_593(CompleteTimeInMillis)> ($ASPO_{ST}$-(d*24*60*60*1000)) ∧ (ID)!=[IDCurr]) .Approve_Shipment_Payment_Order_593(completeResource)
sfo	=	case(current).Approve_Shipment_Payment_Order_593(offerDistribution)

Table 4.1: Variable definition for approval fraud risk.

specified as follows:

$$(ASPO\# > 0) \vee ((sfo_2.\text{startMinNumber} = 0) \wedge (sfo.\text{startMinNumberExcept}.sfo_2 >= 1)). \tag{4.2}$$

An order unfulfillment occurs whenever an order is updated more than five times (see Section 4.1). Accordingly, the respective risk can be detected if: i) task Update Shipment Payment Order is currently being performed for a given order; and ii) it is likely that this order will be updated again (i.e. task Update Shipment Payment Order will be repeated within the same process instance). The probability that the number of times a task will be repeated within the same instance of the Payment subprocess is computed by dividing the number of instances, where five executions for task Update Shipment Payment Order has been reached, over the number of instances that have executed this task at least as many times as it has been executed by the current instance, and have completed. This condition can be checked by using variable $USPO\#US$ to retrieve the amount of orders that have been updated at least as much as this order, and variable $USPO\#U5$ to retrieve the amount of orders that have been updated at least five times.

The variables described can be defined via the sensor definition language as in Table 4.2. Assuming a tolerance value of 60%, the risk condition is specified as follows:

$$(USPO\#U5 / USPO\#US) > 0.6. \tag{4.3}$$

USPO#UC	=	case(current).Update_Shipment_Payment_Order_604(Count)
USPO#U5	=	case(Update_Shipment_Payment_Order_604(Count)>=5). Update_Shipment_Payment_Order_604(CountElements)
USPO#US	=	case(Update_Shipment_Payment_Order_604(Count)>=*USPO#UC* ∧ Process_Shipment_Payment_603(isOffered)="true"). Update_Shipment_Payment_Order_604(CountElements)

Table 4.2: Variable definition for order unfulfillment risk.

An underpayment fraud occurs whenever a customer underpays more than three times in a five-day time frame (see Section 4.1). Accordingly, the respective risk can be detected if: i) task Issue Debit Adjustment is being performed for a given customer and order (this is the trigger for this risk); and ii) the probability that the maximum number of allowable executions for this task will be reached in a five-day time frame, is above the fixed tolerance value for this risk, (this is the risk condition itself). This condition can be checked by using variable *Probability* to retrieve the probability that an attempted fraud will take place. This variable use the *Action* "`FraudProbabilityFunc`" to compute the specific probability (see B), and takes as input, among other information, variable $IDA_{\#\text{Issue}}$ which retrieves the number of times task Issue Debit Adjustment has been completed for this customer.

$IDA_{StartTime}$	=	case(current).Issue_Debit_Adjustment_605(StartTimeInMillis)
c	=	case(current).Issue_Shipment_Invoice_594.ShipmentInvoice.Company
d	=	5
$IDA_{\#Issue}$	=	case(Issue_Shipment_Invoice_594.ShipmentInvoice.Company=*c* ∧ Issue_Debit_Adjustment_605(Count)>0 ∧ Issue_Debit_Adjustment_605 (CompleteTimeInMillis)>($IDA_{StartTime}$-*d**24*60*60*1000)) .Issue_Debit_Adjustment_605(CountElements)
GroupingElement	=	Issue_Shipment_Invoice_594.ShipmentInvoice.Company
WindowElement	=	Issue_Debit_Adjustment_605(CompleteTimeInMillis)
Threshold	=	0.6
Probability	=	case(Issue_Debit_Adjustment_605(Count)>0 ∧ (ID)!=[IDcurr]). Issue_Debit_Adjustment_605(FraudProbabilityFunc, $IDA_{\#Issue}$, 3, *GroupingElement*, *WindowElement*, (*d**24*60*60*1000))

Table 4.3: Variable definition for underpayment fraud risk.

The defined variables are implemented through the sensor language as follows as in Table 4.3. These variables are used to compose the following risk condition, assuming a tolerance value of 60%:

$$Probability > 0.6. \tag{4.4}$$

The definition of actions and functions used in the above variables is provided in appendix. In particular, the complete list of all actions is provided in Ap-

pendix B, the complete lists of the nested loops and of the functions are provided in Appendix C and Appendix D.

4.5 Risk Templates

In this thesis we do not propose techniques for the identification of risk conditions and for mapping such conditions to specific process attributes. These aspects are left to business/risk analysts and can be supported by the use of established methods such as Fault Tree Analysis or Root Cause Analysis. On the other hand, to facilitate the definition of concrete risk conditions for process models, we define the notion of risk template. A risk template is an "abstract" risk condition defined using generic tasks and generic variables associated with these tasks, which are not linked to any real process model. These generic tasks and variables will then be bound to real tasks and variables when the template is used to define a concrete risk condition for a specific process model. These templates are thus used as "shortcuts" to simplify the definition of risk conditions.

Several studies address different types of risks [RM05, CLB04, Meu00, Sch00, Sim99, Sma96]. After an analysis of these studies we decided to subdivide our risk templates into categories reflecting these types of risks. We expect that this division into categories also reduces the effort required by a user to select a suitable template. These categories were defined using the risk taxonomy proposed by Rosemann and zur Muehlen [RM05] and other types of risks proposed in the literature [HBW03]. Our organization of templates based on the above taxonomy is not intended to be an exhaustive coverage of all possible process-related risks. Instead, we provide an extensible structure that can easily be extended based on specific requirements or further taxonomies.

We organized risk templates in two levels. The first level is obtained using the risk taxonomy, obtaining five categories of risks. These categories are: i) *Structural*, capturing risks deriving from wrong decisions taken at design-time; ii) *Data*, capturing risks of damaging data integrity; iii) *Technology*, capturing risks of impacting on the availability of IT systems; iv) *Organizational*, capturing risks of process faults that may impact on the employees or be caused by them; and i) *Goal*, capturing risks of not achieving the process objectives that cannot be categorized in any of the previous categories. For each of these risk categories we defined eleven subcategories. These subcategories are based on the type of risks proposed in the literature [HBW03]. Specifically, we have: i) *Strategic*, risks of affecting the implementation of business strategies; ii) *Operations*, risks of affect-

ing the capability of supplying and producing services or goods; iii) *Supply*, risks that prevent a resource from executing an operation; iv) *Customer*, risks related to customers; v) *Asset*, risks deriving from the use of an asset; vi) *Competitive*, risks deriving from faults related to competitiveness; vii) *Reputation*, risks of loss of reputation; viii) *Financial*, risks of financial loss; ix) *Fiscal*, risks caused by changes in taxation; x) *Regulatory*, risks related to changes in regulations; xi) *Legal*, risks of faults that may lead to legal actions. Figure 4.5 shows a mind map illustrating how risk templates are categorized and subcategorized. This subdivision in categories is not absolute. Thus, a situation in which a risk could belong to more subcategories at the same time is possible. In this case the user should choose the template from the subcategory considered to be the most appropriate, i.e. the one that would generate the risk condition that is the closest to the user requirements.

We defined 14 risk templates as a starting point for a larger risk template repository. For example, one template of type Data/Operations, aims to detect a possible inconsistency in the value of a critical variable. The template requires a as the critical variable, b the identifier variable, and two set of instances s_1 and s_2, where s_1 is composed of the instances in which the values of a and b are equal to the value of a and b for the current instance, and s_2 is composed of the instances in which the value of a is equals to the value of a for the current instance. If the number of instances in s_1 divided by the number of instances in s_2 is greater than a certain threshold the template triggers a notification. This risk template can be used to represent a case in which the critical variable is a credit card number, or a bank account, while the identifier variable is the customer identifier.

Another risk template, of type Goal/Financial, detects the possibility that a particular process may exceed the budget assigned for its execution. This risk is detected looking at all the completed instances of the process and at all the completed instances with an execution cost at least equal to the execution cost of the current instance. The condition calculates the probability that the current instance will exceed the budget. This risk can be associated with any process where the cost is a relevant element. An example is an insurance process where opinions from different assessors may be required but the total cost of their involvement should not exceed the premium.

Considering the category Organizational, subcategory Reputation, we have risks that affect the reputation of the company due to "employees' mistakes". For example, a delivery company may incur in a reputational risk if a delivery needs to be rescheduled due to an employee' mistake. In this case the customer

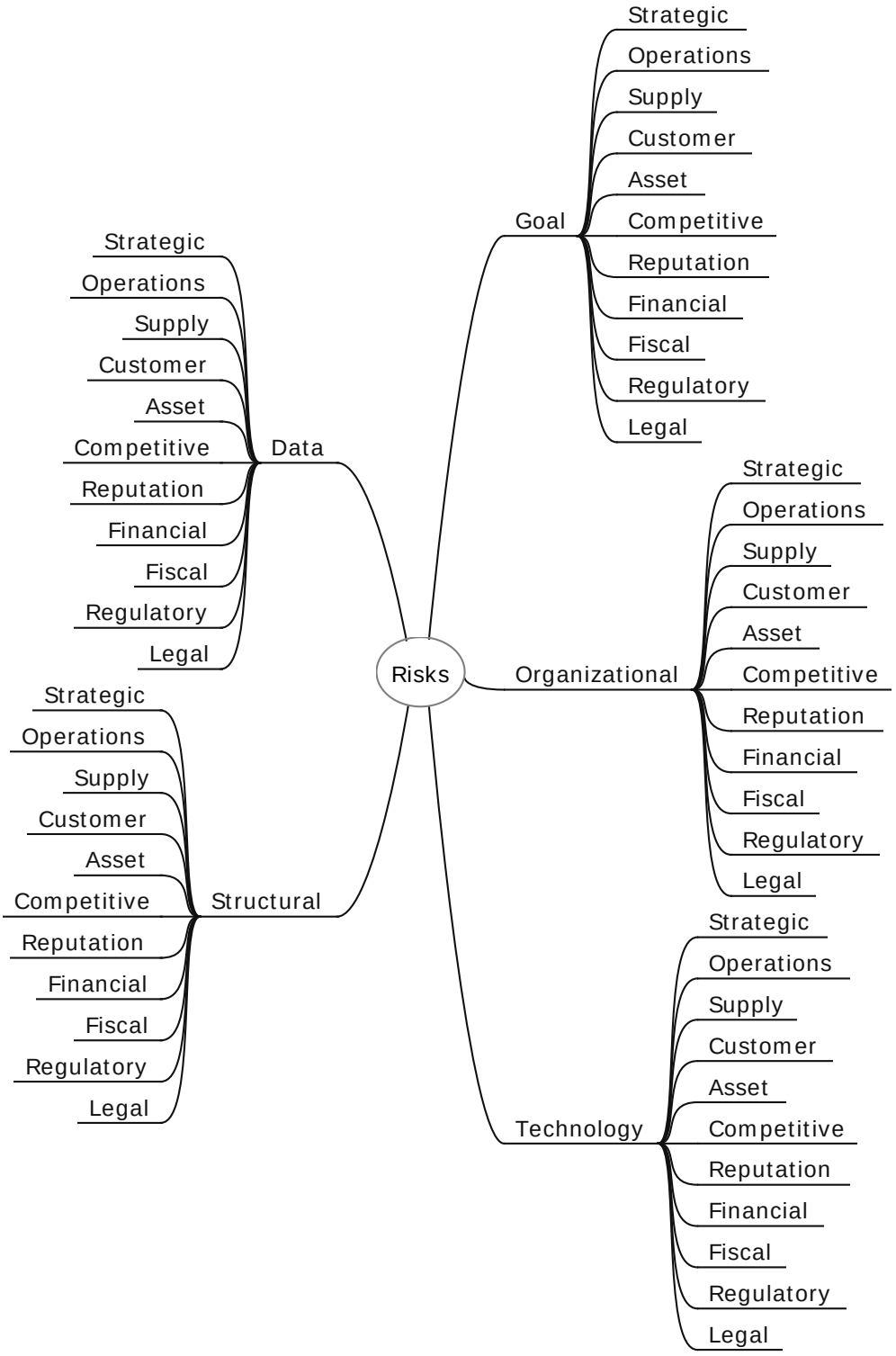


Figure 4.5: Template structure with categories and subcategories.

may think that the company is not professional enough with a consequent impact on the credibility of the company.

In the category Organizational, subcategory Supply, we defined a template that addresses possible delays with the execution of tasks. It detects the possibility that a critical task would not be started as soon as it is offered. This risk is detected looking at all the resources which have been offered a work item of the critical task. If the ratio between the number of resources that are busy versus the total number of resources is greater than a certain threshold the risk is detected. This type of risk is relevant for processes in healthcare such as a process for transferring organs from one hospital to another, where the unavailability of a resource may cause the organ deterioration. Finally, taking in consideration the process described in Section 4.1, let's say that the company wants to avoid that processing a shipment payment may be subjected to delays, what we have to do is: i) import the template; ii) create a mapping of our generic critical task to the task "Process Shipment Payment"; and iii) modify the specified threshold if required.

Table 4.4 provides a brief overview of the templates currently available in the YAWL system (see Section 4.6). Besides the three templates described above, we have, among others, templates for the following risks: underpayment fraud, approval fraud, overtime for activity, sequence of activities and process, and violation of the four-eyes principle within and across cases.

4.6 Software Implementation

In order to prove the feasibility of our technique, we implemented the sensor-based architecture[3] in the YAWL system.

As part of this implementation, we extended the YAWL Editor version 2.2beta with a new component, namely the Sensor Editor, for the specification of sensors within YAWL process models. Such graphical component, shown in Figure 4.6, fully supports the specification of sensor conditions as defined in Section 4.2, and the specification of risk templates.

A wizard (see Figure 4.7) was developed as part of the YAWL Editor to facilitate the use of risk templates and in particular the mapping required for their use. This wizard using an user friendly interface guides the user during each step of the mapping, providing a description of each generic task and variable. The wizard also provides the possibility of customizing a risk, since it is possible

[3]Available at `https://risk-aware-bpm.googlecode.com/`

Category	SubCategory	Template Name	Risk Description
Data	Financial	Underpayment Fraud	Underpayment fraud
	Operational	Data Inconsistency Across Cases	Wrong information provided, detected using multiple cases
		Data Inconsistency Parallel Activity	Wrong information provided to parallel tasks
Goal	Financial	Process Cost Exceeded	Exceeding the budget during the execution of the process
	Strategic	Activity Time Exceeded	Exceeding the time limit within which the activity needs to be completed
		Multi-Activity Time Exceeded	Exceeding the time limit within which a sequence of activities needs to be completed
		Process Time Exceeded	Exceeding the time limit within which the process needs to be completed
Organizational	Financial	Approval Fraud	Approval fraud
	Operational	Four-Eyes Principle	Four-Eyes Principle violation
		Four-Eyes Principle Across Cases	Four-Eyes Principle violation for activities across different cases
		Inadequate Preparation	The activity is executed by a novice resource
	Supply	Delay In Start	The activity start will be delayed
		Task Priority Unfulfilled	Activity with high priority cannot start because resources are busy
Structural	Operational	Loop	A loop is executed too many times

Table 4.4: Risk Templates and descriptions

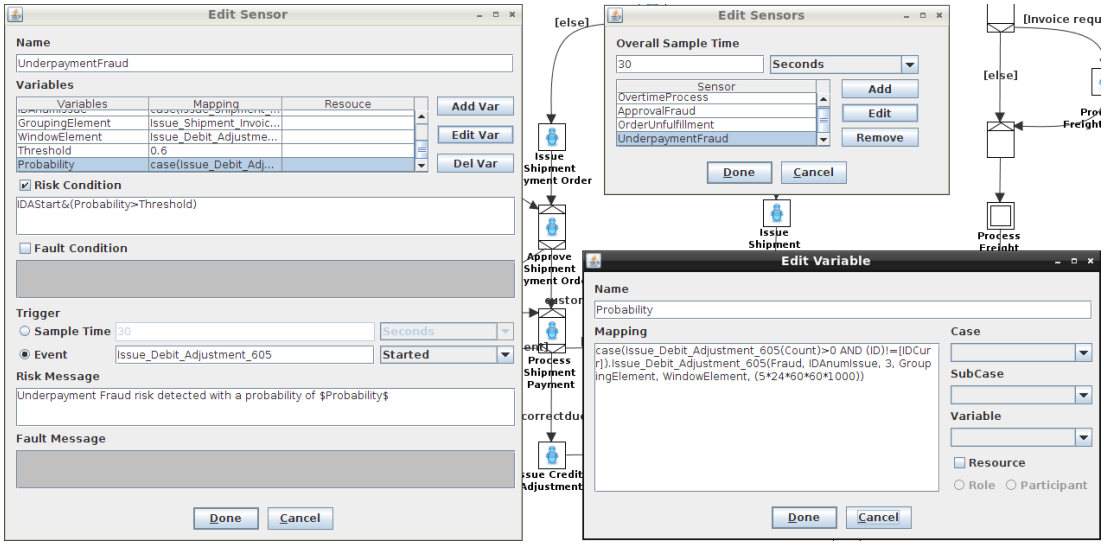


Figure 4.6: The Sensor Editor within the YAWL Editor.

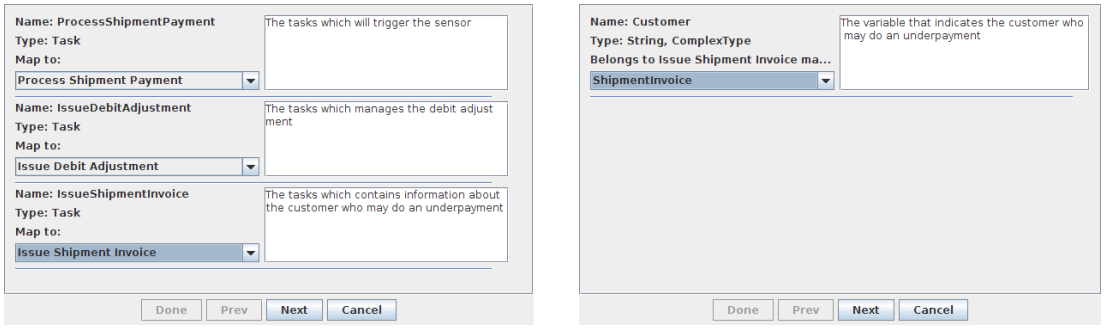


Figure 4.7: Template Wizard: a) Tasks mapping, b) Variables mapping.

to modify a risk condition during the creation of a risk using templates.

Moreover, we implemented the Sensor Manager as a generic component which exposes three interfaces (engine, database and monitor) as described in Section 4.2. We then wrapped this component into a Web service which implements the three interfaces for the YAWL system, allowing the component to interact with the YAWL Engine, the Monitor service and the YAWL database. While there is a straightforward mapping between the YAWL Engine and our engine interface, and between the YAWL Monitor service and our monitor interface, we had to join several YAWL tables to implement our database interface. This is because in the YAWL system, event logs are scattered across different database tables. For example, to retrieve all identifiers of the process instances for a specific process model, given the model identifier, we need to perform a join among the following YAWL tables: `logspecification`, `lognetinstance`, `lognet` and `logevent`.

The complete mapping is illustrated in Table 4.5. As an example, this table also shows the mapping between our database interface and the relational schema used by Oracle BPEL 10g [BZ09] to store BPEL process logs. Also in this case, the database can be fully mapped by joining several tables.

Finally, we implemented a separate service to estimate the remaining cycle time T_e for a process or task instance. This service uses ProM's prediction miner [ASS11] to compute the estimations, and provides the results to the Sensor Manager on demand. While the estimation of T_e could be done on-line, i.e. while evaluating a particular sensor condition at run-time, parsing the full logset each time would be inefficient. Rather, we compute this estimation off-line, whenever a new process model is deployed to the YAWL Engine, by using the logset available at that time. Periodically, we update the logset with the new instances being executed meantime, and invoke this service to refresh the estimations for each process model currently deployed.

Database table	Tables that need to be joined	
	YAWL	**Oracle BPEL 10g**
WorkFlowDefinition	logspecification, lognet, lognetinstance, logevent	cube_instance, cube_scope
SubProcess	logspecification, lognet, lognetinstance, logevent	cube_instance, cube_scope
Activity	lognetinstance, logtask, logtaskinstance, lognet, logevent, logspecification, rs_eventlog	wftask, work_item
Variables	logtask, lognet, lognetinstance, logtaskinstance, logevent, logdataitem, logspecification	audit_trail, audit_detail, xml_document
Role	rs_participant	wftask
ActivityRole	rs_eventlog, logtaskinstance	wftask

Table 4.5: Database interface mapping for YAWL 2.2beta and Oracle BPEL 10g.

4.7 Evaluation

In this section we discuss two evaluations of the sensor-based architecture. In Section 4.7.1, we discuss a performance analysis of the implementation of the architecture in the YAWL system, and in Section 4.7.2, we report on a usability evaluation of the architecture on basis of an online focus-group with 21 BPM practitioners that gained experience in using the system for risk sensor condition analysis and definition tasks.

4.7.1 Performance Analysis

We used our implementation to evaluate the scalability of the approach. First, we measured the time needed to evaluate the basic functions (e.g. counting the number of instances of a task or retrieving the resource allocated to a task). Next, we measured the time needed to evaluate the sensor conditions for the risks defined in the Payment subprocess. The tests were run on an Intel Core I5 M560 2.67GHz processor with 4GB RAM running Linux Ubuntu 11.4. The YAWL logs were stored on the PostGres 9.0 DBMS. These logs contained 318 completed process instances from 36 difference process models, accounting for a total of 9,399 process events (e.g. task instance started and completed, variable's value change). Specifically, there were 100 instances from the Payment subprocess yielding a total of 5,904 process events. The results were averaged over 10 runs.

Table 4.6 shows the results of the evaluation of the basic functions provided by our language. In particular, in this table we compare the evaluation times obtained by accessing the YAWL logs via our database interface, with those obtained by accessing a serialization of the logs, e.g. in the OpenXES format. While OpenXES provides a simple and unique representation of a generic set of process logs, accessing an OpenXES file in real-time, i.e. during the execution of a process instance, is not feasible, due to the long access times (e.g. 6.5 sec. on average for evaluating a net variable). On the other hand, accessing the logs via

Basic function	Description	OpenXES time [ms]	Database time [ms]	Reduction rate [%]
net status	functions checking if a net status has been reached (isStarted, isCompleted)	6,535	18.9	99.71
net time	functions returning the time when a net status has been reached (startTime, completeTime, startTimeInMillis, completeTimeInMillis)	6,781	18.8	99.72
net variable	returns the value of a net variable	6,489	432.6	93.33
task count	number of times a task has been completed	803	19.8	97.53
task resource	functions that return the resources associated with a task (offerResource, allocateResource, startResource, completeResource)	850	20.9	97.54
task status	functions checking if a task status has been reached (isOffered, isAllocated, isStarted isCompleted)	792	30.5	96.14
task time	functions returning the time when a task status has been reached (offerTime, allocateTime, startTime, completeTime, offerTimeInMillis, allocateTimeInMillis, startTimeInMillis, completeTimeInMillis)	824	22.3	97.29
task variable	returns the value of a task variable	787	96.7	87.71
task distribution	functions returning the resources associated with a task by default (offerDistribution, allocateDistribution, startDistribution, completeDistribution)	243		-
task initiator	functions returning the allocation strategy for a resource association (offerInitiator, allocateInitiator, startInitiator, completeInitiator)	249.6		-

Table 4.6: Performance of basic functions.

Sensor	**Min** [ms]	**Max** [ms]	**Ave** [ms]	**St.Dev.**
Overtime process	121	137	131.8	4.66
Approval fraud	6,483	7,036	6,766.4	183.06
Order unfulfillment	69	91	77.4	7.18
Underpayment fraud	3,385	3,678	3,523	89.98

Table 4.7: Performance of sensors.

our database interface, despite it requires the creation of a specific implementation for each BPMS database, provides considerably faster times than accessing OpenXES files (at least 87% gain w.r.t. OpenXES access). In fact, as we can see from Table 4.6, the evaluation times for all the basic functions are below 30 ms, apart from function `task variable`, which takes almost 100 ms and function `net variable`, which takes about 430 ms.

The last two basic functions reported in Table 4.6, namely `task distribution` and `task initiator`, are evaluated in less than 250 milliseconds. These functions are not computed by accessing the logs, but rather by accessing information that is contained directly in an executable process model, e.g. the resources that are associated with a specific task. However, in our implementation we still use the database interface to access this information, in order to provide the developer with a single access point to all process-related data.

Table 4.7 reports the results of the evaluation of the sensor conditions defined for our running example. While the sensor conditions for the overtime process and order unfulfillment faults are very low (below 150 ms), longer times are obtained for evaluating the conditions for the two faults related to fraud (few minut. This is because both these conditions require to evaluate "complex queries", i.e. queries over the entire process logs: in the approval fraud, we need to retrieve all resources that approved an order for a specific customer, while in the underpayment fraud we need to retrieve all process instances where a debit adjustment was issued and aggregate these instances per customer. These queries are different than those needed to evaluate the basic functions, as the latter are performed on the events in the logs that are relative to a single known process instance, e.g. the instance for which the sensor condition is being evaluated.

The worst-case complexity of evaluating one such a complex query is still linear on the number of parameters that need be evaluated in the query (corresponding to the language element *CondExprSet* in Section 4.2) multiplied by the total number of instances present in the logs (corresponding to the size of table WorkflowDefinition addressed by our database interface).

In conclusion, the performances of evaluating sensor conditions should always be considered w.r.t. the specific process for which the risks are defined, and the type of trigger used. For example, let us assume an average duration of 24 hours for the Payment subprocess, with a new task being executed every 30 minutes. This means we have up to 30 minutes to detect an overtime process risk before a new task is executed, and we need to compute this sensor condition again. If we choose a rate of 5 minutes to sample this condition, we are well below the 6 minute-threshold, so we can check this sensor condition up to 6 times during the execution of a task. Since we do this in less than 150 ms, this time is acceptable. For an event-driven risk we also need to consider the frequency of the specific event used as trigger. For example, the approval fraud risk is triggered every time an instance of task Approve Shipment Payment Order is offered to a Senior Financial Officer for execution. Since we take up to 7 seconds to compute this sensor condition, we are able to cope with a system where there is a request for approval every 7 seconds. So also for this sensor, the performance is quite acceptable.

4.7.2 Usability Analysis

For the evaluation of the usability of the sensor-based architecture we followed the example from related studies [LHW+11, LWM+11] and conducted focus group-like online sessions with BPM practitioners in which users had to perform two tasks with the system, viz., analysis of risk conditions, and analysis of risk sensor conditions, and then report on their usage experiences using a structured questionnaire.

Appendix E provides the description of the usability test. The test consisted of five parts:

- In part one, participants had to provide relevant demographic information about their process modeling experience.
- In part two, they were asked to provide details about their experience in using workflow management systems in general.
- In part three, they were asked to provide details about their experience in using YAWL in particular. The questions for parts one to three were, where possible, adapted from prior surveys on process modeling [LHW+11, LWM+11] and usage of the YAWL system [RL12].
- In part four, users were provided with two tasks:

1. analysis of the risk sensor conditions for two YAWL models using the developed risk definition language (risk scenarios 1 and 3 in Appendix E), and
2. definitions of risk sensor conditions for a YAWL model based on the risk definition using fault tree analysis (risk scenario 2 in Appendix E).

We administered these tasks by presenting three risk scenarios on the basis of the developed sensor-based architecture to the users. Risk scenarios 1 and 3 describe risks using the developed risk definition language. Here we asked users to analyze the risk proposed. Risk scenario 2 describes a risk using a fault tree. Here we asked users how such a risk can be defined using the proposed language.

For each of these application tasks, we assessed how well participants were able to assess risks, in terms of *accuracy* of understanding the risks and the *difficulty* of understanding the meaning of the risks. Accuracy and difficulty are widely used measures [BJWW09, MSR12] of the effectiveness and the efficiency of the sensor-based approach in terms of how well users of the approach can understand the risk information provided through the approach, and how much cognitive effort is required to develop this understanding. We used a 5-point scale to measure the difficulty of understanding the risk from very simple to very difficult, and we defined five true/false/do not know questions to measure accuracy of comprehension of the risk information provided.

- In part five, we captured participants' perceptions about the usage experience of the approach as implemented in the YAWL environment through a structured questionnaire. The questionnaire contained measurement items that were adapted from [RL12] and measured satisfaction (SAT), usefulness (PU), ease of use (PEOU) and intentions to use (ITU). Similar to previous uses of these items [Rec10a, Rec10b, RL12], the measures were of appropriate reliability, with Cronbach's Alphas ranging from 0.64 (ITU) to 0.81 (PU).

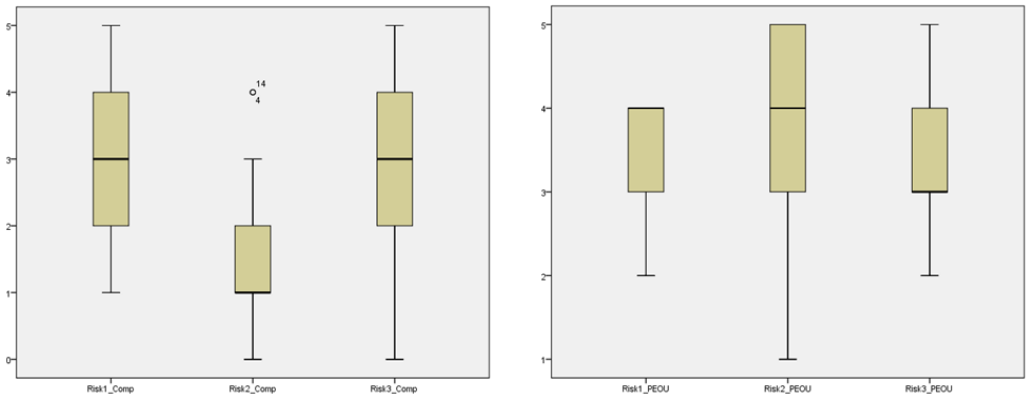
The usability evaluation test was administered through 21 online sessions between January and September 2012. Each online session consisted of a tutorial about the functions and use of the sensor-based architecture, which that consisted of an introduction, the creation of risk sensors in the YAWL editor and detailed instructions for defining sensors from scratch or on the basis of the templates.

Additional documentation provided descriptions and explanations of actions, resource functions and loops relevant to risk sensors. After completing the tutorial, each participant was asked to conduct the usability evaluation by completing the tasks above administered through a structured online questionnaire. The sessions were not timed and no time limits were imposed on the users. Since these sessions were not timed no information about their duration were collected.

For the usability evaluation, we recruited participants by contacting individuals that had previously completed a university under-graduate or post-graduate course on business process automation. The cohorts were from the Universitá della Calabria, Cosenza, Italy and the Queensland University of Technology, Brisbane, Australia, and we perused the course email lists for contact. We recruited participants from these cohorts because principles of process modeling and automation in general as well as the YAWL system specifically were featured in the courses, thus allowing us to select participants with sufficient levels of background experience and expertise, in turn minimizing the risk of confounding the usability results due to lack of experience with workflow systems or process modeling in general.

Overall twenty-one participants participated, where 10 of them were from Italy and 11 from Australia. Participants, on average, had about 2.6 years of experience with process modeling and had read and created, on average, 71 and 15 process models, respectively, over the last twelve months. Participants' experience with different process modeling languages varied, with most having experience, at least, with UML Activity Diagrams, Petri Nets, EPCs and/or BPMN. These characteristics describe our participants as proxies for novice to average BPM professionals, with one of our participants representative of an expert practitioners (more than 5 years experience, more than 250 models created, more than 1000 models read). Overall, our study population might not be representative of experts but is roughly equivalent to the range of expertise found in actual BPM practitioners as reported in other surveys [Rec10c].

The box-plots in Figure 4.8a and Figure 4.8b show the accuracy of the comprehension of the risks in the three scenarios as well as the perceived difficulty of understanding these risks. The data shows that scenarios 1 and 3 were reasonably well understood (averages for the comprehension questions were 2.9 for scenario 1 and 3.1 for scenario 3, both exceeding 50% accuracy), while comprehension of scenario 2 was significantly lower (average score 1.7). Similarly, perceived difficulty was highest for scenario 2 (average of 3.7), while the perceived difficulty for scenario 1 and 3 was similar in range (averages 3.4 and 3.3, respectively).



(a) Accuracy of risk comprehension, on a scale from 0 (low) to 5 (high).

(b) Difficulty of risk comprehension, on a scale from 1 ("very simple") to 5 ("very difficult").

Figure 4.8: Accuracy and difficulty of risk comprehension.

Next we were interested in understanding under which circumstances participants were able to obtain higher levels of accuracy in understanding the risk information provided. To that, we built a regression model in which we regressed relevant demographic data and perceived difficulty onto the total comprehension score across all three risk scenarios. Specifically, we included as coefficients:

- PMExp(Years): Process modeling experience in years
- PMExp(Training): Extent of formal training in process modeling in days over the last twelve months
- PMExp(SelfEducation): Extent of self-education in process modeling in days over the last twelve months
- PMExp(processModelsCreated): Number of process models created over the last twelve months
- BPMSFAM: Self-perceived familiarity with BPMSs [Rec10a]
- YAWLUse(time): length of use of the YAWL system
- YAWLUse(features): Number of YAWL features used
- YAWLUse(models): Number of YAWL models created, read or edited over the last twelve months
- RISKPerDif: Average perceived comprehension difficulty across the three presented scenarios.

Variable	Standardized Coefficients		
	Beta	t	Sig.
PMExp(Years)	**-0.80**	**-2.79**	**0.02**
PMExp(Training)	-0.03	-0.16	0.88
PMExp (processModelsCreated)	**1.28**	**4.31**	**0.00**
BPMSFAM	0.38	1.15	0.28
YAWLUse(time)	**0.80**	**3.07**	**0.01**
YAWLUse(features)	-0.30	-1.40	0.20
YAWLUse(models)	**-1.20**	**-3.86**	**0.00**
RISKPerDif	-0.19	-1.05	0.32

Table 4.8: Regression analysis of risk comprehension across all three scenarios

The regression model was significant ($F = 3.54$, $p = 0.04$) and explained 77.9 percent of the variance in total comprehension score, thus attesting to very good explanatory power. Table 4.8 gives the results from the regression model analysis.

Table 4.8 shows that four factors were significant predictors for explaining comprehension accuracy, these being process modeling experience in years, number of process models created, use of the YAWL system and number of YAWL models created, edited or read over the last twelve months. Several interesting findings are noteworthy. First, PMExp(Years) and YAWLUse(models) are negative predictors, which may suggest that novice users with little process modeling experience and little experience with YAWL models benefited more from the risk sensor approach. Second, the difficulty of understanding the risk information provided is not related to how well users understand the risks. Third, training in process modeling or more varied use of the YAWL system, likewise, are irrelevant in terms of understanding risk information provided by the sensor-based architecture.

Finally, we were interested in the participants' evaluation of the sensor-based architecture. Following the theory of technology acceptance [Dav89] and its extended application in the process modeling context [Rec10b], we understand that satisfaction, ease of use and perceived usefulness are key criteria for explaining intentions to use a process modeling artifact. Figure 4.9 shows the box-plots for the average total factor scores for these four criteria.

The data displayed in Figure 4.9 suggests that participants rated the usefulness of the approach high (mean score > 5.3) and were in general inclined to use the system (mean score > 4.4). Satisfaction with the approach was reported as average (mean score $= 4.0$) and ease of use, notably, was reported as low (mean score < 3.5), indicating potential to improve interface and user interaction of the

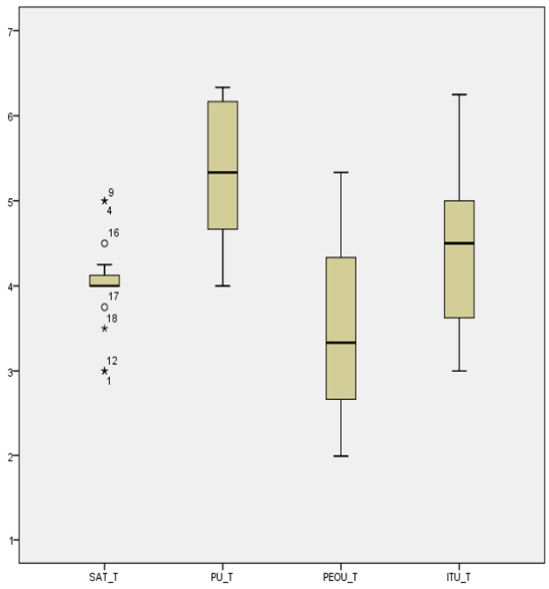


Figure 4.9: Perceptual evaluations of the sensor-based architecture on a scale from 1 (low) to 7 (high).

system. This is not surprising, since we did not explicitly consider the principle of "clarity" when designing the concrete syntax of our language.

In a post-hoc analysis, we compared evaluations of SAT, PU, PEOU, and ITU across users that scored low or high on risk comprehension, based on a median split, and across users that rated the perceived difficulty of understanding risks as low or high, again based on a median split. MANOVA tests in both cases showed that evaluations of the approach were independent from comprehension accuracy or difficulty, with p-values ranging from 0.30 to 0.49 and 0.18 to 0.58, respectively. The results indicate that the evaluations of satisfaction, usefulness, ease of use and intentions to use were robust against variances in the performance in using the system.

In summary, our evaluation showed that the sensor-based risk identification and modeling approach provided value in allowing participants to develop an understanding of process risks. We found the approach to be useful particularly for users with limited experience in process modeling or BPMSs. The evaluation further revealed that ease of use of the system should be improved to warrant better user acceptance.

4.8 Related Work

From the architectural point of view, our technique, specifically our sensor-based architecture, shares commonalities with Oracle Business Activity Monitoring (BAM), that we presented in Chapter 2. Oracle BAM relies on sensors to monitor the execution of BPEL processes. Three types of sensors can be defined: activity sensors, to grab timings and variable contents of a specific activity; variable sensors, to grab the content of the variables defined for whole BPEL process (e.g. the inputs to the process); and fault sensors, to monitor BPEL faults. These sensors can be triggered by a predefined set of events (e.g. task activation, task completion). For each sensor, one can specify the endpoints where the sensor will publish its data at run-time (e.g. a database or a JMSQueue). Compared to this approach we allow the specification of more sophisticated sensor (and fault) conditions, where different process-related aspects can be incorporated such as data, resource allocation strategies, order dependencies, as well as historical data and information from other running process instances. Moreover, our sensors can be triggered by process events or sampled at a given rate. Nonetheless, our sensor-based architecture is exposed as a service and as such it could be integrated with other process monitoring systems, such as Oracle BAM.

A *Sensor*, as defined in this work, may be categorized as a specialization of the more general rules framework known as *Event-Condition-Action* (ECA) rules, which originally came to prominence through their use with Active Database Management Systems [DBM88]. A Sensor extends the ECA notion by including two conditions, one which describes a potential *fault*, and the other the possibility of *risk*. A Sensor also delivers a consequence when a triggered condition evaluates to true, rather than an action *per se*. ECA rules have also been used as the basis for several well-known exception handling strategies in workflow systems [Cas99, CIJ+00, CLK99]. For example, a definition language for exception rules called *Chimera-Exec* was developed for the *WIDE* prototype [CFM99]. An extended ECA framework can be found in the *defeasable* workflow approach [LSKA03], which has an added *justification* condition to support context-dependent reasoning of actions to be taken. The *RUMBA* project combines ECA rules with an Aspect-Oriented framework to provide a modular approach to the integration of rules and active processes [CAS06]. The *Worklet* approach incorporates an extended rules-based framework, which embeds exceptions directly into the evaluation tree [AHAE07].

Real-time monitoring of process models can also be achieved via Complex

Event Processing (CEP) systems. In this context, CEP systems have been integrated into commercial BPMSs, e.g. webMethods Business Events[4], ARIS Process Event Monitor [DB07] and SAP Sybase [Syb11], as well as explored in academia [GPL+10, HSD10]. A CEP system allows the analysis of aggregated events from different sources (e.g. databases, email accounts as well as process engines). Using predefined rules, generally defined with a specific SQLlike language [WREW11], a CEP system can verify the presence of a specific pattern among a stream of simple events processed in a given time window. Our technique differs from CEP systems in the following aspects: i) strong business process orientation vs general purpose system; ii) ability to aggregate and analyze complex XML-based events (e.g. process variables) vs simple events; iii) time-driven and event-driven triggers vs event-driven trigger only. Moreover, CEP systems typically suffer from performance overheads [HSD10, WREW11] which limit their applicability to real-time risk detection [WREW11].

4.9 Summary

In this chapter we contributed a technique for real-time monitoring of risks in executable business process models. The technique embeds elements of risk into each phase of the BPM lifecycle: from process design, where high-level risks defined via a risk analysis method are mapped down to specific process model elements such as activities, resources and data, through to process diagnosis, where risks are detected during process execution, and those no longer tolerable are notified to process administrators.

As a second contribution, we provided an operationalization of the proposed technique on top of BPMSs. This is achieved via a distributed, sensor-based architecture that communicates with a BPMS via a set of tool-independent interfaces. Each risk is associated with a sensor condition and refers to a fault, which is an undesired state of the process. Conditions can relate to any process aspect, such as control-flow dependencies, resource allocations, the content of data elements, both from the current process instance and from instances of any process that have already been completed. At design-time, these conditions are expressed within a process model via a simple query language, for which we provide an abstract syntax. At run-time, each sensor independently alerts a sensor manager when the associated risk condition evaluates to true during the execution of a specific process instance. When this occurs, the sensor manager notifies

[4]http://www.softwareag.com/au/products/wm/events/overview

a process administrator about the given risk by interfacing with the monitoring service of the BPMS. This allows early risk detection which in turn enables proper remedial actions to be taken in order to avoid potentially costly process faults.

We designed a set of risk templates to allow process designers to easily specify new risk conditions into a process model. Each template captures an abstract risk. To use these templates, one has to bind the template variables to concrete elements of the process model for which the risk condition needs to be monitored. We contend that by using such templates the effort of defining risks in executable process models can be reduced.

As a proof-of-concept, we implemented the sensor-based architecture on top of the YAWL system along with 14 representative templates. We then used the tool to evaluate the feasibility of the approach in practice. This was carried out in two directions. First, we evaluated the performance of the implementation; second, we evaluated the usability and ease of use of the approach with users of the YAWL system. The performance measurements showed that the sensor conditions can be computed efficiently and that no performance overhead is induced to the BPMS engine. The results of the empirical evaluation with users showed that the sensor-based risk identification and modeling approach provided value to develop an understanding of process risks. We found the approach to be useful particularly for users with limited experience in process modeling or BPMSs. The evaluation further revealed that ease of use of the language for defining sensors should be improved to warrant better user acceptance.

The technique presented in this chapter and its operationalization serve as the cornerstone for our risk-aware business process management approach. In the next chapter we will present a technique which builds on the approach presented in this chapter to allow process participants to make risk-informed decisions. This is achieved by predicting the risk that the current process instance will end up with one or more faults based on the input provided by the participant, e.g. when filling out a user form based on the conditions specified for each fault.

Chapter 5

Risk Prediction

In the previous chapter as part of our approach for risk-aware business process management we presented a technique to model risks in executable business process models and detect them as early as possible during process execution. However, the limitation of these efforts is that risks are not *predicted*, but only *detected* when their likelihood exceeds a tolerance threshold, hence faults cannot be prevented.

To address these limitations here we present a recommendation system that supports process participants in taking risk-informed decisions, with the aim of *reducing* process risks preemptively. A process participant takes a decision whenever they have to choose the next task to execute out of those assigned to them at a given process state, or via the data they enter in a user form. This input from the participant may influence the risk of a process fault to occur. For each such input, the technique returns a risk prediction in terms of the likelihood and severity that a fault will occur if the process instance is carried out using that input. This prediction is obtained via decision trees which are trained using historical process data such as process variables, resources, task durations and frequencies. The historical data of a process is observed using decision trees which are built from the execution logs of the process, as recorded by the IT systems of an organization.

This way, the participant can take a risk-informed decision as to which task to execute next, or can learn the predicted risk of submitting a form with particular data. If the instance is subjected to multiple potential faults, the predictor can return the weighted sum of all fault likelihoods and severities, as well as the individual figures for each fault. The weight of each fault can be determined based on the severity of the fault's impact on the process objectives.

The above technique only provides "local" risk predictions, i.e. predictions

relative to a specific process instance. In reality, however, multiple instances of (different) business processes may be executed at any time. Thus, we need to find a risk prediction for a specific process instance that does not affect the prediction for other instances. The interplay between risks relative to different instances can be caused by the sharing of the same pool of process participants: two instances may require the same scarce resource. In this setting, a sub-optimal distribution of process participants to the set of tasks to be executed, may result in a risk increase (e.g. overtime or cost overrun risk). To solve this problem, we equipped our recommendation system with a second technique, based on integer linear programming, which takes input from the risk prediction technique, to find an *optimal distribution* of process participants to tasks. By optimal distribution we mean one that minimizes the overall execution time (i.e. the time taken to complete *all* running instances) while minimizing the overall level of risk. This distribution is used by the system to suggest process participants the next task to perform.

We operationalized our recommendation system on top of the YAWL system by extending an existing YAWL plug-in and by implementing two new custom YAWL services. This implementation prompts process participants with risk predictions upon filling out a form or for each task that can be executed. We then evaluated the effectiveness of our system by conducting experiments using a claims handling process in use at a large insurance company. With input from a team of risk analysts from the company, this process has been extensively simulated on the basis of a log recording one year of completed instances of this process. The recommendations provided by our system significantly reduced the number and severity of faults in a simulation of a real life scenario, compared to the process executed by the company as reflected by the event data. Further, the results show that it is feasible to predict risks across multiple process instances without impacting on the execution performance of the BPM system.

The remainder of this chapter is organized as follows. Section 6.1 presents the running example. Next, Section 5.2 defines the notions of event logs and faults which are required to explain our techniques. Section 5.3 describes the technique for predicting risks in a single process instance while Section 5.4 extends this technique to the realm of multiple process instances running concurrently. Section 5.5 and Section 5.6 discuss the implementation and evaluation of the overall technique, respectively. Finally, Section 5.7 discusses related work before Section 5.8 concludes the chapter.

The research presented in this chapter has been the subject of several publi-

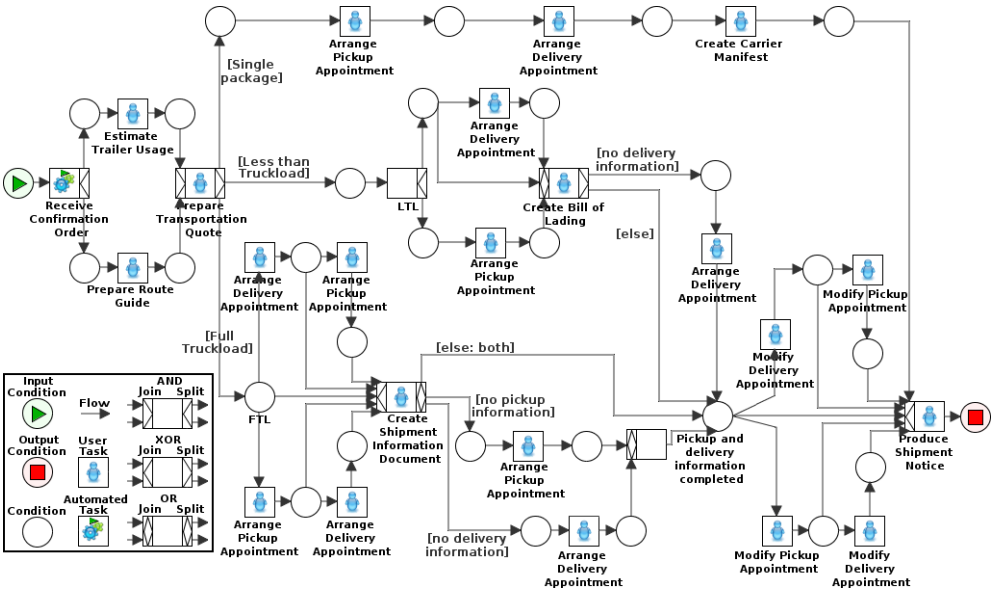


Figure 5.1: The YAWL model of the Carrier Appointment subprocess of the Order Fulfillment process mentioned in Chapter 4.

cations [CLLA13, CLH+13, CLL+14].

5.1 Running Example

We developed our technique on top of the YAWL language (see Chapter 3). In this section, we present the Carrier Appointment subprocess of the Order Fulfillment process mentioned in Chapter 4. The model of this process, using the YAWL language, is shown in Figure 5.1. In the Order Fulfillment process, the Payment subprocess proceeds the Payment subprocess and it is as the latter subjected to several risks.

The Carrier Appointment subprocess (see Figure 5.1) starts when a Purchase Order Confirmation is received. A Shipment Planner then estimates the trailer usage and prepares a route guide. Once ready, a Supply Officer prepares a quote for the transportation indicating the cost of the shipment, the number of packages and the total freight volume.

If the total volume is over 10,000 lbs a full trackload is required. In this case two different Client Liaisons will try to arrange a pickup appointment and a delivery appointment. Before these two tasks are performed, a Senior Supply Officer may create a Shipment Information document. In case the Shipment Information document is prepared before the appointments are arranged, a Warehouse Officer will arrange a pickup appointment and a Supply Officer will arrange a delivery appointment, with the possibility of modifying these appointments until a Ware-

house Admin Officer produces a Shipment Notice, after which the freight will be picked up from the Warehouse.

If the total volume is up to 10,000 lbs and there is more than one package, a Warehouse Officer arranges the pickup appointment while a Client Liaison may arrange the delivery appointment. Afterwards, a Senior Supply Officer creates a Bill of Lading, a document similar to the Shipment Information. If a delivery appointment is missing a Supply Officer takes care of it, after which the rest of the process is the same as for the full trackload option.

Finally, if a single package is to be shipped, a Supply Officer has to arrange a pickup appointment, a delivery appointment, and create a Carrier Manifest, after which a Warehouse Admin Officer can produce a Shipment Notice.

5.2 Fault Severity

Each completed trace of the event log is assigned a fault's severity between 0 and 1, where 0 identifies an execution with no fault and 1 identifies a fault with the highest severity. To model this, a risk analyst needs to provide a fault function f. The set of all such functions is:

$$\mathcal{F} = (T \times R \times \mathbb{N} \times \Phi)^* \to [0, 1]$$

In many settings, processes are associated with different faults. These faults can be combined together by assigning different weights. Let us suppose to have n faults $\{f_1, \ldots, f_n\} \subset \mathcal{F}$, we can have a *composite fault*:

$$\widehat{f}(\sigma) = \frac{\sum_{1 \leq i \leq n} w_i f_i(\sigma)}{\sum_{1 \leq i \leq n} w_i} \in \mathcal{F}$$

where w_i is the weight of the fault f_i, with $1 \leq i \leq n$.

A complete trace σ of our Carrier Appointment process, can be affected by three faults:

Over-time fault. This fault is linked to a Service Level Agreement (SLA) which establishes that the process must terminate within a predefined Maximum Cycle Time d_{mct} (e.g. 21 hours), in order to avoid pecuniary penalties that will incur as consequence of a violation of the SLA. The severity of the fault grows with the amount of time that the process execution exceeds d_{mct}. Let d_σ be the duration of the process instance, i.e. difference between the timestamps of the last and first event of σ. Let $d_{\max}$ be the maximum

duration among all process instances already completed (including σ). The severity of an overtime fault is measured as follows:

$$f_{time}(\sigma) = \max\left(\frac{d_\sigma - d_{mct}}{\max(d_{\max} - d_{mct}, 1)}, 0\right)$$

Reputation-loss fault. During the execution of the process when a "pickup appointment" or a "delivery appointment" is arranged, errors with location or time of the appointment may occur due to a misunderstanding between the company's employee and the customer. In order to keep the reputation high, the company wants to avoid these misunderstandings and having to call the customer again. The severity of this fault is:

$$f_{rep}(\sigma) = \begin{cases} 0 & \text{if tasks } \textit{Modify Delivery Appointment} \text{ and } \textit{Modify Pick-up Appointment} \text{ do not appear in } \sigma \\ 1 & \text{if both } \textit{Modify Delivery Appointment} \text{ and } \textit{Modify Pick-up Appointment} \text{ appear in } \sigma \\ 0.5 & \text{otherwise} \end{cases}$$

Cost Overrun fault. During the execution of this process, several activities need to be executed, and each of these has an execution cost associated with it. Since the profit of the company decreases with a higher shipping cost of a good (or goods), the company wants to reduce them. Of course, there is a profit cost beyond which the company will not make any profit. The severity increases as the cost goes beyond the profit cost. Let $c_{\max}$ be the greatest cost associated with any process instance that has already been completed (including σ). Let c_σ be the cost of σ and $c_{\min}$ be the profit cost. The severity of a cost fault is:

$$f_{cost}(\sigma) = \min\left(\frac{\max(c_\sigma - c_{\min}, 0)}{\max(c_{\max} - c_{\min}, 1)}, 1\right)$$

Moreover, we assume that the company considers Reputation-loss Fault to be less significant than the other faults. The company could decide to define a composite fault where the reputation weights half:

$$f_{car}(\sigma) = \big(f_{cost}(\sigma) + f_{time}(\sigma) + 0.5 \cdot f_{rep}(\sigma)\big)/2.5$$

The risk is the product of the estimation of the fault's severity at the end of the process-instance execution and the accuracy of such an estimation.

When a process instance is being executed, many factors may influence the risk and, ultimately, the severity of a possible fault. For instance, a specific order in which a certain set of tasks is performed may increase or decrease the risk, compared to any other. Nonetheless, it is opportune to leave freedom to resources to decide the order of their preference. Indeed, there may be factors outside the system that let resources opt for a specific order. For similar reasons, when there are alternative tasks that are all enabled for execution, a risk-aware decision support may highlight those tasks whose execution yields less risk, anyway leaving the final decision up to the resource.

5.3 Risk Estimation

We aim to provide work-items' recommendation to minimize the risk corresponding to the highest product of fault severity and likelihood. For this purpose, it is necessary to predict the most likely fault severity associated with continuing the execution of a process instance for each enabled task. The problem of providing such a prediction can be translated into the problem of finding the best estimator of a function.

Definition 7 (Function estimator)**.** *Let $X_1, \ldots, X_n$ be n finite or infinite domains. Let Y be a finite domain. Let $f : X_1 \times X_2 \times \ldots \times X_n \rightarrow Y$. An estimator of function f is a function $\psi_f : Y \rightarrow 2^{X_1 \times X_2 \times \ldots \times X_n \times [0,1]}$, such that, for each $y \in Y$, $\psi_f(y)$ returns a set of tuples $(x_1, \ldots, x_n, l)$ where $(x_1, \ldots, x_n) \in (X_1 \times X_2 \times \ldots \times X_n)$ is an input domain tuple for which the expected output is y and l is the accuracy of such an estimation. Moreover, $(x_1, \ldots, x_n, l_1) \in \psi_f(y_1) \wedge (x_1, \ldots, x_n, l_2) \in \psi_f(y_2) \Rightarrow l_1 = l_2 \wedge y_1 = y_2$.*

The function estimator is trained through a set of observation instances. An observation instance is a pair $(\overrightarrow{x}, y)$ where $\overrightarrow{x} \in X_1 \times X_2 \times \ldots \times X_n$ is the observed input and $y \in Y$ is the observed output.

The function estimator can easily be built using a number of machine learning techniques. In this paper, we employ the C4.5 algorithm to build decision trees. We decided to use decision tree classification, and specifically the C4.5 algorithm, for the following reasons: i) it can handle both continuous and discrete (categorical) attributes; ii) it can handle training data with missing attribute values; iii) it can build models that can be easily interpreted; iv) it can deal with noise; and v) it automatically finds a subset of the features that are relevant to the classification (i.e. no need for feature selection).

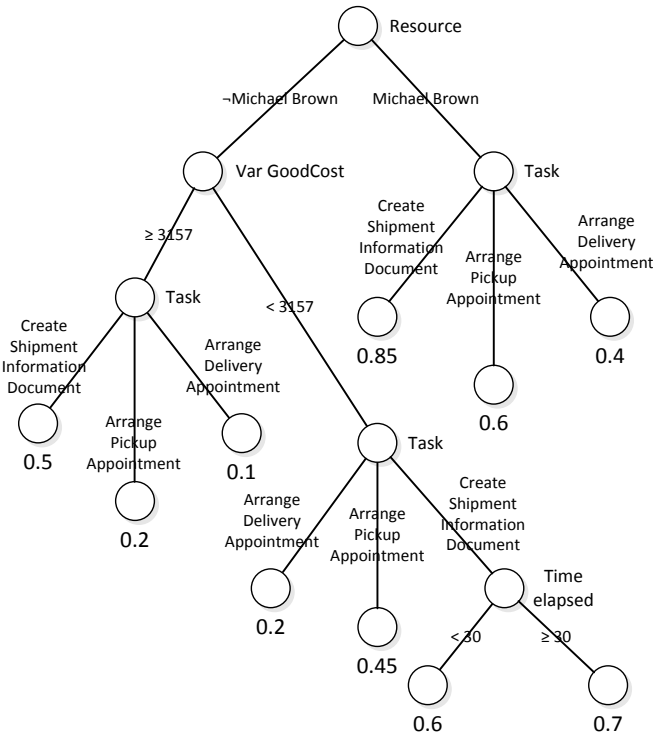


Figure 5.2: An example of decision tree used to build a function estimator.

Decision trees classify instances by sorting them down in a tree from the root to some leaf node. Each non-leaf node specifies a test of some attribute $x_1, \ldots, x_n$ and each branch descending from that node corresponds to a range of possible values for this attribute. In general, a decision tree represents a disjunction of conjunctions of expressions: each path from the tree root to a leaf corresponds to an expression that is, in fact, a conjunction of attribute tests. Each leaf node is assigned one of the possible output values: if an expression e is associated with a path to a leaf node $\overline{y}$, every tuple $\overrightarrow{x} \in X_1 \times X_2 \times \ldots \times X_n$ satisfying e is expected to return $\overline{y}$ as output.

We link the accuracy of a prediction for $\psi_f(\overline{y})$ to the quality of e as classifying expression. Let I be the set of observation instances used to construct the decision tree. Let $I_e = \{(\overrightarrow{x}, y) \in I \mid \overrightarrow{x} \text{ satisfies } e\}$ and $I_{e,\overline{y}} = \{(\overrightarrow{x}, y) \in I_e \mid y = \overline{y}\}$. The accuracy is $l = |I_{e,\overline{y}}|/|I_e|$; therefore, for all $((x_1, \ldots, x_n), y) \in I_e, (x_1, \ldots, x_n, l) \in \psi_f(\overline{y})$. Figure 5.2 shows an example of a possible decision tree. It is the estimator $\psi_{f_{\hat{c}}}$ of a function that returns a value belonging to the set H containing the numbers between 0 and 1 with no more than 2 decimals. It is obtained through a set of observation instances based on all data attributes generated during the execution of the process. For example, having as data attributes a resource, a task, the cost of a good, and a process instance's elapsed time, we obtained the following

function $f_{\hat{c}} : \mathit{Resource} \times \mathit{Task} \times \mathit{GoodCost} \times \mathit{TimeElapsed} \rightarrow H$. For instance, let us consider the value $y = 0.6$. Analyzing the tree, the value is associated with two expressions: e_1 is $(R = \mathit{MichaelBrown} \wedge T = \mathit{ArrangePickupAppointment})$ and e_2 is $(R \neq \mathit{MichaelBrown} \wedge \mathit{GoodCost} < 3157 \wedge \mathit{TimeElapsed} < 30 \wedge T = \mathit{CreateShipmentInformationDocument})$. Let us suppose that, among observation instances $(\mathit{Resource}, \mathit{Task}, \mathit{GoodCost}, \mathit{TimeElapsed}, y)$ s.t. e_1 or e_2 evaluates to true, $y = 0.6$ occurs 60% or 80% of times, respectively. Therefore, $\psi_{f_{\hat{c}}}(0.6)$ contains the tuples $(R, T, \mathit{GoodCost}, \mathit{TimeElapsed}, 0.6)$ satisfying e_1, along with tuples$(R, T, \mathit{GoodCost}, \mathit{TimeElapsed}, 0.8)$ satisfying e_2. Regarding computational complexity, if decision trees are used, training ψ_f with m observation instances is computed in quadratic time with respect to the dimension n (i.e. the number of attributes) of the input tuple, specifically $O(n^2 \cdot m)$ [Qui93].

As mentioned before, it is necessary to predict the most likely fault severity associated with continuing the execution of a process instance with each task enabled for execution. Function estimators are used for such a prediction.

Let $N = (T_N, C_N, i, o, F_N, R_N, V_N, U_N, \mathit{can}_N, \mathit{Split}, \mathit{Join}, \mathit{UserTaskPriv})$ be a YAWL net. In order to provide accurate risks associated with performing work items of a certain process instance, it is important to incorporate the execution history of that process instance into the analysis. In order to avoid overfitting predictive functions the history needs to be abstracted. Specifically, we abstract the execution history as two functions: $C_r : T_N \rightarrow R$ denoting the last executor of each task and $C_t : T_N \rightarrow \mathbb{N}$ denoting the number of times that each task has been performed in the past. Pairs $(c_r, c_t) \in C_r \times C_t$ are called *contextual information*. Given the execution trace of a (running) instance $\sigma' \in (T_N \times R_N \times \mathbb{N} \times \Phi)$, we introduce function $\mathsf{getContextInformation}(\sigma')$ that returns the contextual information (c_r, c_t) that can be constructed from σ'.

Let Φ be the set of all possible assignments of values to variables, i.e. the set of all partial functions $V_N \not\rightarrow U_N$. Each condition $c \in C_N$ can be associated with a function $f_c : \Phi \times c^\bullet \times R_N \times \mathbb{N} \times C_r \times C_t \rightarrow H$. If $f_c(\phi, t, r, n, c_r, c_t) = y$, at the end of the execution of the process instance, the fault's severity is going to be y if the instance continues with resource $r \in R_N$ that performs task $t \in c^\bullet$ at time n with contextual information (c_r, c_t) when variables are assigned values as for function ϕ. Of course, this function is not known but it needs to be estimated, based on the behavior observed in an event log $\mathcal{L}$. Therefore, we need to build am estimator ψ_{f_c} for f_c. Let us consider condition c_{FTL} (see Figure 5.1), and the associated function estimator $\psi_{f_{c_{FTL}}}$. Let us suppose that the accuracy is 1, i.e. for each $t \in {c_{FTL}}^\bullet$, $\psi_{f_{c_{FTL}}}(t)$ always returns 1.

Algorithm 1: generateFunctionEstimatorsForRiskPrediction

Data: $\mathbf{N} = (T_N, C_N, i, o, F_N, R_N, V_N, U_N, can_N)$ – A YAWL net, $\mathcal{L}$ – An event log, $\mathfrak{f} \in \mathcal{F}$ – A fault function

Result: A Function Ψ that associates each condition $c \in C_N$ with a function estimator ψ_c

Let I be a function whose domain is the set of conditions $c \in C_N$, and initially for all $c \in C_N$, $I(c) = \emptyset$.

foreach trace $\sigma = \langle (t_1, r_1, d_1, \phi_1), \ldots, (t_n, r_1, d_n, \phi_n) \rangle \in \mathcal{L}$ **do**
  Set function A such that $\mathsf{dom}(A) = \emptyset$
  for $i \leftarrow 1$ **to** n **do**
    $(c_r, c_t) \leftarrow \mathsf{getContextInformation}(\langle (t_1, r_1, d_1, \phi_1), \ldots, (t_i, r_i, d_i, \phi_i) \rangle)$
    Time elapsed $\overline{d} \leftarrow (d_i - d_1)$
    $J \leftarrow (A \odot (t_i, r_i, \overline{d}) \odot c_r \odot c_t), \mathfrak{f}(\sigma))$
    foreach $c \in {}^\bullet t_i$ **do**
      $I(c) \leftarrow I(c) \cup \{J\}$
    end
    foreach *variable* $v \in \mathit{dom}(\phi_i)$ **do**
      $A(v) \leftarrow \phi_i(v)$
    end
  end
end
Set function Ψ such that $\mathsf{dom}(\Psi) = \emptyset$
foreach *condition* $c \in C_N$ **do**
  $\Psi(c) \leftarrow \mathsf{buildFunctionEstimator}(I(c))$
end
return Ψ

If the execution is such that there is a token in *FTL*, *GoodCost* < 3157, executing tasks *Arrange Pickup Appointment*, *Arrange Delivery Appointment* are associated with a risk of 0.2 and 0.45, respectively. Conversely, executing task *Create Shipment Information Document* is given a risk of either 0.6 or 0.7, depending on the moment in which task *Create Shipment Information Document* is started. Therefore, it is evident that it is less "risky" to execute *Arrange Pickup Appointment*.

Algorithm 1 details how function estimators ψ_{f_c} can be constructed. In the algorithm, we use $\odot$ to concatenate tuples: given two tuples $\overrightarrow{x} = (x_1, \ldots, x_n)$ and $\overrightarrow{y} = (y_1, \ldots, y_m)$, $\overrightarrow{x} \odot \overrightarrow{y} = (x_1, \ldots, x_n, y_1, \ldots, y_m)$. Operator $\odot$ can also be overloaded to deal with functions defined on a finite and ordered domain. Let $f : W \to Z$ be a function defined on an ordered domain $W = \{w_1, \ldots, w_o\}$. If we denote $z_i = f(w_i)$ with $1 \leq i \leq o$, $f \odot \overrightarrow{x} = (z_1, \ldots, z_o, x_1, \ldots, x_n)$.

Algorithm 1 is periodically executed, e.g., every week or after every k process instances are completed. In this way, the predictions are updated according to the recent process executions. The input parameters of the algorithm are a YAWL net N, an event log with traces referring to past executions of instances of the process modelled by N, and a fault function. The output is a function Ψ that

associates each condition c with function estimator ψ_{f_c}. Initially, in line 1, we initialize function I which is going to associate each condition c with the set of observation instances associated with the executions of tasks in the postset of p. From line 2 to line 12, we iteratively replay all traces σ to build the observation instances. While replaying, a function A keeps the current value's assignment to variables (line 3). For each trace's event (t_i, r_i, d_i, ϕ_i), first we build the tuple (c_r, c_t) of the contextual information (line 5) and compute the elapsed time $\overline{d}$ (line 6). Then, we build an observation instance J where tuple $(A \odot (t_i, r_i, \overline{d}) \odot c_r \odot c_t)$ is the observed input and the fault severity $f(\sigma)$ is the observed output. This observation instance is put into the set of observation instances relative to each condition $c \in {}^\bullet t_i$. In lines 11-13, we update the current value's assignment during the replay, i.e. we rewrite function A. Finally, in lines 16-19, we build each function estimator ψ_{f_c} for condition f_c by the relative observation instances and rewrite Ψ s.t. $\Psi(c) = \psi_c$.

In this section, we presented a technique to generate prediction functions. It is important to observe that the number of risks that may eventuate during the execution of a process does not affect the prediction algorithm, since we consider the combined risk level of all risks. Specifically, we do so by assigning a relative weight to each risk. This weight system allows process administrators to fine tune the predictive function on the basis of the relative importance of each risk.

5.4 Multi-Instance Work-Item Distribution

With the technique presented so far, each resource is given *local* risk advice as to what work item to perform next, i.e. a resource is suggested to perform the work item with the lowest overall risk for that combination of process instance and resource, without looking at other resources that may be assigned work items within the same instance or in other instances running concurrently. Clearly, such a local work-item distribution is not optimal, since work items have to compete for resources and this may not guarantee the best allocation from a risk viewpoint. For example, let us consider two resources r_1 and r_2 and two work items w_a and w_b such that the risk of r_1 performing w_a is 0.2, and the risk of r_1 performing w_b is 0.6, while the risk of r_2 performing w_a is 0.1 and the risk of r_2 performing w_a is 0.4. Moreover for the company executing these work items, it is equally important to minimize the eventuation of risks as well as the overall execution time. If w_a is assigned to r_2 because locally this resource has the lowest risk, r_1 will be forced to perform w_b leading to an overall risk of 0.7. Another option is to

assign both work items to r_2, yielding an overall risk of 0.5. Both these solutions are non-optimal distributions: the former because the overall risk is too high, the latter, despite the lower risk, because the workload between the two resources is unbalanced, with the result of increasing the overall execution time.

In this section we combine our technique for risk prediction with a technique for computing an *optimal distribution* of work items to resources (available or busy). By optimal distribution we mean a distribution that minimizes the weighted sum of overall execution time and overall risk across all running instances. In other words, the algorithm aims to balance the distribution of work items across resources while keeping the risk low. This distribution can then be used to provide work item recommendations to resources, such that these can be aided in selecting the best work item to perform. In the example above, the optimal distribution is r_1-w_a and r_2-w_b with an overall risk of 0.6. While this is higher than 0.5 obtained with the second solution, r_1 and r_2 will work in parallel thus reducing the overall execution time.

5.4.1 Optimal Work-Item Distribution

Let $\mathfrak{f}$ be a certain (composite) fault function and assuming we at time τ. Let $I = \{id_1, \ldots, id_n\}$ be the set of running instances of N. Given an instance $id \in I$, $timeElapsed_\tau(id) \in \mathbb{N}$ denotes the time elapsed since instance id has started and $varAssign_\tau(id) \in (V_N \to U_N)$ is the current assignment of values to variables. Moreover, let us denote a function $use_N : R_N \to 2^{T_N \times I}$ that associates each resource with the work items that he/she is executing within the set I of running process instances. Let WE be the set of work items being executed, i.e. $WE = \sum_{r \in R_N} use_N(r)$. Let $W \subseteq T_N \times I$ be the set of work items that are enabled but not started yet. Section 6.1 has discussed the concept of deferred choice, highlighting that some of the enabled work items are mutually exclusive. Therefore, we introduce an equivalence relation $\sim$ between elements of W, such that $w_a \sim w_b$ if, picking $w_a \in W$ for execution disables $w_b \in W$ or vice versa. Let $W_\sim$ be the partition of W according to relation $\sim$.

For each enabled work item $w \in W$, we perform an estimation $time_w$ of the expected duration of work item w. For each started work item $w \in WE$, we also perform an estimation $time_w$ of the amount of time needed by w to be completed. To compute such estimations, we employ the technique proposed by Van der Aalst et al. [ASS11] using event log $\mathcal{L}$ as input.

Let Ψ be the set of function estimators that are computed through Algorithm 1, using net N, event log $\mathcal{L}$ and given fault function $\mathfrak{f}$ as input.

Algorithm 2: calcRisk

Data: $\mathbf{N} = (T_N, C_N, i, o, F_N, R_N, V_N, U_N, can_N)$ – A YAWL net, $f \in \mathcal{F}$ – A fault function, r – resource, t – time, (ta, id) – work item
Result: A risk value
$risk \leftarrow 0$
$\phi \leftarrow varAssign(id)$
$d \leftarrow timeElapsed(id)$
$(c_r, c_t) \leftarrow \mathsf{getContextInformation}(history(id))$
foreach condition $c \in {}^\bullet t$ **do**
  $\psi \leftarrow \Psi(c)$
  Pick $(severity, l)$ such that $(\phi, t_a, r, d, c_r, c_t, l) \in \psi(severity)$
  $risk \leftarrow \max(severity \cdot l, risk)$
end

For each work item $w \in W$, let us denote with $risk_{r,w,t}$ the risk of starting a work item w at time t. This can be computed by invoking Algorithm 2: $risk_{r,w,t} = \mathsf{calcRisk}(N, \mathfrak{f}, r, t, w, \Psi)$.

Let $maxTime = \sum_{w \in W \cup WE} time_w$ be the maximum duration of executing all work items that are currently enabled and started. This corresponds to the situation in which work items are just executed sequentially, i.e. a new work item starts only when no other work item is being executed. Given a resource $r \in R_N$ and a work item $(ta, id) \in W$ such that $ta \in can_N(r)$, we compute the set of moments in time in which the risk of r performing (ta, id):

$$init_{r,w} = \{t \in [\tau, \tau + maxTime] \mid risk_{r,w,t} \neq risk_{r,w,t-1}\} \cup \{\tau\}$$

Certainly, this can be naively computed by computing the risk for all moments in time between τ and $\tau + maxTime$. Nonetheless, it can be done more efficiently by observing the occurrences of splits on the time variable that are present in the decision trees. For instance, let us consider the decision tree in Figure 5.2: the only time reference is 30. This reference occurs in a root-to-leaf path in which resource $r \neq$ *Michael Brown* and $Task =$ *Create Shipment Information*. Therefore, for each resource $r \in R \setminus \{$*Michael Brown*$\}$ and work-item $w =$ (*Create Shipment Information*, id) $\in W$, $init_{r,w} = \{\tau, elapsed(id) + 30\}$. Moreover, for each work item $w = (ta, id) \in W$ with $ta \neq$ *Create Shipment Information* and for each resource $r \in R$, $init_{r,w} = \{\tau\}$. Similarly, for each work item $w = (ta, id) \in W$, $init_{r',w} = \{\tau\}$ with $r' =$ *Michael Brown*.

Given a work item w, a resource r and a time t, $\Delta_{r,w}(t)$ denotes the first moment t' in time after t in which the risk changes, i.e. $t' > t$, $t' \in init_{r,w}$ and there exists no $t'' \in init_{r,w}$ such that $t' > t'' > t$. If such a moment t' does not exist, $\Delta_{r,w}(t) = \tau + maxTime$.

We formulate the problem of distributing work items as a Mixed-Integer Linear Programming (MILP) problem. The following two sets of variables are introduced:

- For each resource $r \in R_N$ and work-item $w = (ta, id) \in W$ such that $ta \in can_N(r)$, there exists a variable $x_{r,w,t}$. If the solution of the MILP problem is such that $x_{r,w,t} = 1$, r is expected to start performing w in interval between t and $\Delta_{r,w}(t)$, $x_{r,w,t} = 1$; otherwise, $x_{r,w,t} = 0$;

- For each work item $w \in W \cup WE$ (i.e., running or enabled), we introduce a variable $a_{r,w}$. If work item w is not being executed at time τ and is eventually distributed to resource r, the MILP solution assigns to $a_{r,w}$ a value that is equal to the moment in time when resource r is expected to start work item w. If w is not expected to be started by r, $a_{r,w} = 0$; if w is already being executed by r at time τ (i.e. $w \in WE$), $a_{r,w}$ is statically assigned value τ.

The MILP problem aims to minimize the weighted sum of the expected total execution time and the overall risk:

$$\min \left(\frac{\alpha}{maxTime} \sum_{r \in R_N} \sum_{w \in W \cup WE} a_{r,w} + (1-\alpha) \sum_{r \in R_N} \sum_{w \in W \cap can_N(r)} \sum_{t \in init_{r,w}} risk_{r,w,t} \cdot x_{r,w,t} \right)$$

where $\alpha \in [0, 1]$ is the weight of the expected total execution time w.r.t. the overall risk.

This MILP problem is subject to a number of constraints:

- For each $r \in R_N$ and $w = (ta, id) \in W$ such that $ta \in can_N(r)$, if r starts performing w in the interval between t and $\Delta_{r,w}(t)$, $x_{r,w,t}$ must be equal to 1 (and vice versa):

$$x_{r,w,t} = 1 \Leftrightarrow \Delta_{r,w}(t) > a_{r,w} \wedge a_{r,w} \geq t; \tag{5.1}$$

- For each partition $D \in W_{\sim}$, only one work item in D can be executed and it can only be executed by one resource and can only start within one interval:

$$\sum_{r \in R_N} \sum_{w \in D \cap can_N(r)} \sum_{t \in init_{r,w}} x_{r,w,t} = 1 \tag{5.2}$$

- Every resource $r \in R_N$ cannot execute more than one work item at any time.

Therefore, for each $r \in R_N$ and for each pairs of partitions $D_1, D_2 \in W_\sim$:

$$\begin{aligned}\Big(\sum_{w_a \in D_1} a_{r,w_a} - \sum_{w_b \in D_2} a_{r,w_b} \geq \sum_{w_b \in D_2} \sum_{t \in init_{r,w_b}} time_{w_b} \cdot x_{r,w_b,t}\Big) \vee \\ \Big(\sum_{w_b \in D_2} a_{r,w_b} - \sum_{w_a \in D_1} a_{r,w_a} \geq \sum_{w_a \in D_1} \sum_{t \in init_{r,w_a}} time_{w_a} \cdot x_{r,w_a,t}\Big)\end{aligned} \tag{5.3}$$

The constraints in Equations 5.1 can be translated into an equivalent set of linear constraints as follows:

$$\begin{aligned}&-a_{r,w} - M \cdot (1 - x_{r,w,t}) \leq -t \\ &a_{r,w} - M \cdot (1 - x_{r,w,t}) < \Delta_{r,w}(t) \\ &a_{r,w} - M \cdot x_{r,w,t} - M \cdot o'_{r,w,t} < t \\ &-a_{r,w} - M \cdot x_{r,w,t} - M \cdot (1 - o'_{r,w,t}) \leq -\Delta_{r,w}(t)\end{aligned} \tag{5.4}$$

where M is a sufficiently large number (e.g., the largest machine-representable number) and $o_{r,w,t}$ is a boolean variable that needs to be introduced in the MILP problem.

Lemma 1. *Constraints of the form as in Equations 5.1 can be rewritten into sets of equivalent constraints of the form as in Equations 5.4.*

Proof.

Let us consider $x_{r,w,t}$ and its possible values 1 and 0. If $x_{r,w,t} = 1$ then the last two constraints will be satisfied by $-M \cdot x_{r,w,t} \ll t - a_{r,w}$ and $-M \cdot x_{r,w,t} \ll -\Delta_{r,w}(t) - a_{r,w}$. In order to satisfy the first two constraints, since $M \cdot (1 - x_{r,w,t}) = 0$, $a_{r,w}$ must be $a_{r,w} \geq t \wedge a_{r,w} < \Delta_{r,w}(t)$, that is exactly the second part of the constraint defined in Equations 5.1.

If $x_{r,w,t} = 0$ then $M \cdot (1 - x_{r,w,t}) = M$. This satisfies the first two constraints since $-M \cdot (1 - x_{r,w,t}) \ll -t + a_{r,w}$ and $-M \cdot (1 - x_{r,w,t}) \ll \Delta_{r,w}(t) - a_{r,w}$. The third constraint can be satisfied only if $a_{r,w} < t$ or if $o'_{r,w,t} = 1$, similar thing can be said for the fourth constraint that will be satisfied only if $a_{r,w} \geq \Delta_{r,w}(t)$ or if $o'_{r,w,t} = 0$. We can derive that in order to satisfy the last two constraints we either have $a_{r,w} < t$ and $o'_{r,w,t} = 0$, or we have $a_{r,w} \geq \Delta_{r,w}(t)$ and $o'_{r,w,t} = 1$. As we can see for $x_{r,w,t} = 0$ the only way to satisfy the constraints of Equations 5.4 is to violate the second part of the constraint defined in Equations 5.1. □

Similarly, the constraints in Equation 5.3 can be transformed into a set of

linear constraints as follows:

$$\begin{aligned}
&\sum_{w_b \in D_2} a_{r,w_b} - \sum_{w_a \in D_1} a_{r,w_a} + \sum_{w_b \in D_2} \sum_{t \in init_{r,w_b}} time_{w_b} \cdot x_{r,w_b,t} - M \cdot o_{r,D_1,D_2,t} \leq 0 \\
&\sum_{w_a \in D_1} a_{r,w_a} - \sum_{w_b \in D_2} a_{r,w_b} + \sum_{w_a \in D_1} \sum_{t \in init_{r,w_a}} time_{w_a} \cdot x_{r,w_a,t} - M \cdot (1 - o_{r,D_1,D_2,t}) \leq 0
\end{aligned} \tag{5.5}$$

where M is a sufficiently large number and $o_{r,D_1,D_2,t}$ is a boolean variable that needs to be introduced in the MILP problem.

Lemma 2. *Constraints of the form as in Equations 5.3 can be rewritten into sets of equivalent constraints of the form as in Equations 5.5.*

Proof. Let us consider the constraints in Equations 5.5, and let introduce for readability purposes the following equality:

$$\begin{aligned}
&\sum_{w_b \in D_2} a_{r,w_b} - \sum_{w_a \in D_1} a_{r,w_a} + \sum_{w_b \in D_2} \sum_{t \in initrw_b} time_{w_b} \cdot x_{r,w_b,t} = a \\
&\sum_{w_a \in D_1} a_{r,w_a} - \sum_{w_b \in D_2} a_{r,w_b} + \sum_{w_a \in D_1} \sum_{t \in initrw_a} time_{w_a} \cdot x_{r,w_a,t} = b.
\end{aligned} \tag{5.6}$$

we can then rewrite Equations 5.5 as:

$$\begin{aligned}
a - M \cdot o_{r,D_1,D_2,t} \leq 0 \\
b - M \cdot (1 - o_{r,D_1,D_2,t}) \leq 0
\end{aligned} \tag{5.7}$$

The first constraint in Equations 5.5 can only be satisfied if either $a \leq 0$ or if $-M \cdot o_{r,D_1,D_2,t} \leq 0$. Similarly, the second constraint can only be satisfied if either $b \leq 0$ or if $-M \cdot (1 - o_{r,D_1,D_2,t}) \leq 0$. Since $o_{r,D_1,D_2,t}$ can only be 0 or 1, we can see that in order to satisfy both constraints either $a \leq 0$ or $b \leq 0$ must be satisfied that is exactly the constraint defined in Equations 5.3. □

As an example of an instance of the class of MILP problems, let us consider a case where at time τ we want to schedule three work items w_a, w_b and w_c, and we have two resources, r_1 and r_2, who can perform them. We know that w_a and w_b are mutually exclusive generating the following partitions $D_1 = \{w_a, w_b\}$, and $D_2 = \{w_c\}$. Moreover, we know that the expected duration of each work item is $time_{w_a} = 30$ mins, $time_{w_b} = 10$ mins, and $time_{w_c} = 40$ mins. We also know that the risk associated with each work item does not change over time. Finally, we know that when performed by resource r_1 the work items have the following expected risk levels: $risk_{r_1,w_a,\tau} = 0.2$, $risk_{r_1,w_b,\tau} = 0.7$, and $risk_{r_1,w_c,\tau} = 0.6$ while when performed by resource r_2 the work items have the following expected risk levels: $risk_{r_2,w_a,\tau} = 0.1$, $risk_{r_2,w_b,\tau} = 0.7$, and $risk_{r_2,w_c,\tau} = 0.4$.

The MILP problem for distributing work items will take the following form (assuming $\alpha = 0.5$):

$$\begin{aligned}\text{minimize } & \frac{0.5}{\tau + 80} \cdot (a_{r_1,w_a} + a_{r_1,w_b} + a_{r_1,w_c} + a_{r_2,w_a} + a_{r_2,w_b} + a_{r_2,w_c}) \\ & + 0.5 \cdot (0.2 \cdot x_{r_1,w_a,\tau} + 0.7 \cdot x_{r_1,w_b,\tau} + 0.6 \cdot x_{r_1,w_c,\tau} + 0.1 \cdot x_{r_2,w_a,\tau} \\ & + 0.7 \cdot x_{r_2,w_b,\tau} + 0.4 \cdot x_{r_2,w_c,\tau})\end{aligned}$$

subject to the following constraints:

either work item w_a or w_b is executed, whereas w_c has to (instantiation of Equation 5.2):

$$x_{r_1,w_a,\tau} + x_{r_1,w_b,\tau} + x_{r_2,w_a,\tau} + x_{r_2,w_b,\tau} = 1$$

$$x_{r_1,w_c,\tau} + x_{r_2,w_c,\tau} = 1$$

at any time, all resources, i.e. r_1 and r_2, can only perform one work item (instantiation of Equation 5.3):

$$\left(a_{r_1,w_c,-} a_{r_1,w_a} - a_{r_1,w_b} \geq 30 \cdot x_{r_1,w_a,\tau} + 10 \cdot x_{r_1,w_b,\tau}\right) \vee$$

$$\left(a_{r_1,w_a} + a_{r_1,w_b} - a_{r_1,w_c} \geq 40 \cdot x_{r_1,w_c,\tau}\right)$$

$$\left(a_{r_2,w_c} - a_{r_2,w_a} - a_{r_2,w_b} \leq 30 \cdot x_{r_2,w_a,\tau} + 10 \cdot x_{r_2,w_b,\tau}\right) \vee$$

$$\left(a_{r_2,w_a} + a_{r_2,w_b} - a_{r_2,w_c} \leq 40 \cdot x_{r_2,w_c,\tau}\right)$$

instantiation of Equation 5.1 for resources r_1, r_2 and work items w_a, w_b, w_c:

$$x_{r_1,w_a,\tau} = 1 \Leftrightarrow a_{r_1,w_a} \geq \tau \wedge a_{r_1,w_a} < \tau + 80$$

$$x_{r_1,w_a,\tau} = 1 \Leftrightarrow a_{r_2,w_a} \geq \tau \wedge a_{r_2,w_a} < \tau + 80$$

$$x_{r_1,w_b,\tau} = 1 \Leftrightarrow a_{r_1,w_b,\geq}\tau \wedge a_{r_1,w_b,<}\tau + 80$$

$$x_{r_1,w_b,\tau} = 1 \Leftrightarrow a_{r_2,w_b} \geq \tau \wedge a_{r_2,w_b} < \tau + 80$$

$$x_{r_1,w_c,\tau} = 1 \Leftrightarrow a_{r_1,w_c,\geq}\tau \wedge a_{r_1,w_c,<}\tau + 80$$

$$x_{r_1,w_c,\tau} = 1 \Leftrightarrow a_{r_2,w_c} \geq \tau \wedge a_{r_2,w_c} < \tau + 80$$

The optimal solution to this problem is $a_{r_1,w_a} = 1$, $a_{r_1,w_b} = 0$, $a_{r_1,w_c} = 0$, $a_{r_2,w_a} = 0$, $a_{r_2,w_b} = 0$, $a_{r_2,w_c} = 1$, $x_{r_1,w_a,\tau} = 1$, $x_{r_1,w_b,\tau} = 0$, $x_{r_1,w_c,\tau} = 0$, $x_{r_2,w_a,\tau} = 0$, $x_{r_2,w_b,\tau} = 0$, $x_{r_2,w_c,\tau} = 1$, that is a schedule where resource r_1 performs work item w_a and resource r_2 performs work item w_c.

5.4.2 Recommendations for Work Items Execution

After the optimal distribution is computed, we need to provide a recommendation to r for executing any $w \in W \cap can_N(r)$. For any work item w, the recommendation $rec(w, r)$ is a value between 0 and 1, where 0 is assigned to the work item with the highest recommendation and 1 to the work item with the least one. Let us consider an optimal solution s of the MILP problem to distribute work items while minimizing risks. The work-item recommendations for each resource r are given as follows:

- If there exists a work item $w \in W \cap can_N(r)$ such that $x_{r,w,\tau} = 1$ for solution s, the optimal distribution suggests w to be performed by r at the current time. Therefore, $rec(w, r) = 0$. For any other work item w', the value $rec(w', r)$ is strictly greater than 0 and lower than or equal to 1:

$$rec(w', r) = \frac{risk_{r,w',\tau} + risk_{r,w,\tau}}{risk_{r,w,\tau} + 1}$$

  $rec(w', r)$ grows proportionally to $risk_{r,w',\tau}$, with $rec(w', r) = 1$ if $risk_{r,w',\tau} = 1$.

- Otherwise, r is supposed to start no work item at the current time. However, since recommendations need to be provided also to resources that are not supposed to execute any work item, for each $w \in W \cap can_N(r)$, we set $rec(w, r) = risk_{r,w,\tau}$.

It is possible that the optimal distribution assigns no work item to a resource r at the current time. This is the case when r is already performing a work item (i.e., no additional work item should suggested) or there are more resources available than work items to assign.

Let us consider the problem illustrated at the end of Section 5.4.1. In this problem we have two resources r_1 and r_2 and three work items w_a, w_b, and w_c. We recall that the expected risk levels associated with a resource performing a given work item were: $risk_{r_1,w_a,\tau} = 0.2$, $risk_{r_1,w_b,\tau} = 0.7$, and $risk_{r_1,w_c,\tau} = 0.6$ for resource r_1, and $risk_{r_2,w_a,\tau} = 0.1$, $risk_{r_2,w_b,\tau} = 0.7$, and $risk_{r_2,w_c,\tau} = 0.4$ for resource r_2. We can then derive that the best allocation requires that resource r_1 performs work item w_a and resource r_2 performs work item w_c. Finally, when recommendations about which work item should be performed and by whom will they be required, the system will return the following values: $rec(r_1, w_a) = 0$, $rec(r_1, w_b) = 0.75$ and $rec(r_1, w_c) = 0.67$ for resource r_1, and $rec(r_2, w_a) = 0.36$, $rec(r_2, w_b) = 0.79$ and $rec(r_2, w_c) = 0$ for resource r_2.

5.4.3 Recommendations for Filling Out Forms

In addition to providing risk-informed decision support when picking work items for execution, we provide support during the execution of the work items themselves. Human resources usually perform work items by filling out a form with the required data. The data that are provided may also influence a process risk. Therefore, we want to highlight the expected risk whenever a piece of data is inserted by the resource into the form.

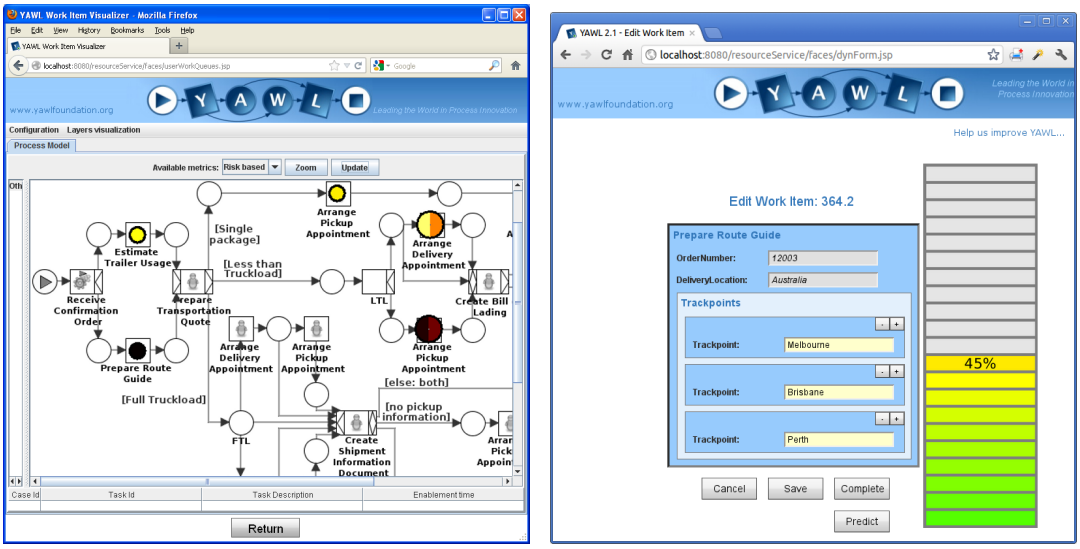
The risk associated with filling a form with particular data is also computed using Algorithm 2. When used to compute the risk associated with filling a form to perform a work item (ta, id), $varAssign(id)$ is the variable assignment that would result by submitting a form using the data the resource has inserted so far.

5.5 Software Implementation

We operationalized our recommendation system[1] on top of the YAWL BPM system, by extending an existing YAWL plug-in and by implementing two new custom YAWL services. This way we realized a risk-aware BPM system supporting multi-instance work distribution and forms filling-out.

The intent of our recommendation system is to "drive" participants during the execution of process instances. This goal can be achieved if participants can easily understand the suggestions proposed by our tool. For this we decided to extend a previous plug-in for the YAWL Worklist Handler, named *Map Visualizer* [LAAH12]. This plug-in provides a graphical user interface to suggest process participants the work items to execute, along with assisting them during the execution of such work items. The tool is based on two orthogonal concepts: maps and metrics. A *map* can be a geographical map, a process model, an organizational diagram, etc. For each map, work items can be visualized by dots which are located in a meaningful position (e.g., for a geographic map, work items are projected onto the locations where they need to be executed, or for a process-model map onto the boxes of the corresponding tasks in the model). Dots can also be colored according to certain *metrics*, which determine the suggested level of priority of a work item. This approach offers advantages over traditional BPM systems, which are only equipped with basic client applications where work items available for execution are simply enlisted, and sorted according to given criteria. When users are confronted with hundreds of items, this visualization

[1] Available at `https://risk-aware-bpm.googlecode.com/`



(a) The UI to support participants in choosing the next work item to perform based on risks.

(b) The UI to support participants in filling out a form based on risks.

Figure 5.3: Screenshots of the Map Visualizer extension for risk-aware prediction in YAWL.

does not scale well. The validity of the metaphors of maps and metrics used for decision support in process execution was confirmed through a set of experiments reported by de Leoni et al. [LAAH12]. They only define very basic metrics. We have extended the repertoire of these metrics with a new metric that is computed by employing the technique described in Section 5.4.

Figure 5.3a shows a screenshot of the Map Visualizer where a risk-based metric is employed. The map shows the process model using the YAWL notation and dots are projected onto the corresponding elements of the model. Each dot corresponds to a different work item and is colored according to the risks for the three faults defined before. When multiple dots are positioned on the same coordinates, they are merged into a single larger dot whose diameter grows with the number of dots being amalgamated. Colors go from white to black, passing through intermediate shades of yellow, orange, red, purple and brown. The white and black colors identify work items associated with a risk of 0 and 1, respectively. The screenshot in Figure 5.3a refers to a configuration where multiple process instances are being carried out at the same time and, hence, the work items refer to different process instances. The configuration of dots highlights that the risk is lower if the process participant performs a work item of task *Estimate Trailer Usage*, *Arrange Pickup Appointment* or *Arrange Delivery Appointment* for a certain instance. When clicking on the dot, the participant is shown the process

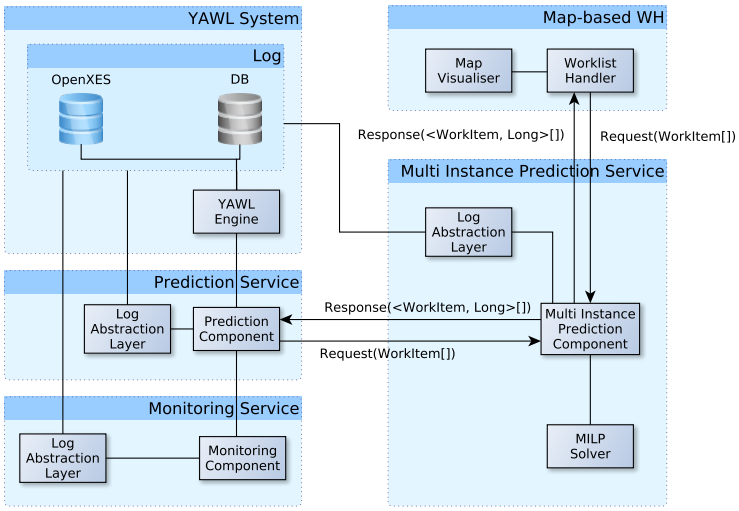


Figure 5.4: The integration of the risk predictor tool with the YAWL system.

instance of the relative work item(s).

As discussed in Section 5.4.3, the activity of compiling a form is also supported. Figure 5.3b shows a screenshot where, while filling in a form, participants are shown the risk associated with that specific input for that form via a vertical bar (showing a value of 45% in the example, which means a risk of 0.45). While a participant changes the data in the form, the risk value is recomputed accordingly.

Besides the extension to the Map Visualizer, we implemented two new custom services for YAWL, namely the *Prediction Service* and *Multi Instance Prediction Service*. The *Prediction Service* provides risk prediction and recommendation. It implements the technique described in Section 5.3 and constructs decision trees through the implementation of the C4.5 algorithm of the Weka toolkit for data mining.[2] Since the algorithm is not capable of predicting continuous values, in order to provide a risk prediction we grouped risk levels that are close to each other in intervals of 0.05 (e.g. all risk likelihoods from 0 to 0.04 are considered as 0, from 0.05 to 0.09 as 0.1 and so on).

The Prediction Service communicates with the *Log Abstraction Layer* described in Chapter 4, to be able to retrieve event logs from textual files, such as from OpenXES event logs, or directly from the YAWL database, which stores both historical information and the current system's state. Moreover it communicates with the *Monitoring Service* in order to figured out which instances completed with a fault and which one did not.

[2]The Weka toolkit is available at `http://www.cs.waikato.ac.nz/ml/weka/`

The *Multi Instance Prediction Service*, similarly to the *Prediction Service*, provides risk prediction and recommendation. The difference between these two services is that in the former a recommendation takes into account *all* process instances currently running in the system. The Multi Instance Prediction Service interacts with the Prediction Service to obtain "local" predictions that, in combination with other information derived from the log (e.g. expected task duration, other running instances), are used to find the optimal resource allocation using the technique described in Section 5.4. To this purpose, the Multi Instance Prediction Service also interacts with the MILP Solver. The *MILP Solver* provides an interface for the interaction with different integer linear programming solvers. So far we support Gurobi[3], SCIP[4] and LPSolve[5]. Finally, the Multi Instance Prediction Service is invoked by the *Map Visualizer* to obtain the risk predictions and recommendations and show these to process participants in the form of maps. The map visualizer works with the standard Worklist Handler provided by YAWL to obtain the up-to-date distribution of work to resources. Figure 6.2 shows the diagram of these connections.

5.6 Evaluation

We evaluated our technique using the claims handling process and related event data, of a large insurance company kept under condition of anonymity. The event data recording about one year of completed instances (total: 1,065 traces) was used as a benchmark for our evaluation. The claims handling process, modeled in Figure 5.5, starts when a new claim is received from a customer. Upon receipt of a claim, a file review is conducted in order to assess the claim, then the customer is contacted and informed about the result of the assessment. The customer may provide additional documents ("Receive Incoming Correspondence"), which need to be processed ("Process Additional Information") and the claim may need to be reassessed. After the customer has been contacted, a payment order is generated and authorized in order to process the payment. During the execution of the process model, several updates about the status of the claim may need to be provided to the customer as follow-ups. The claim is closed once the payment has been authorized.

As one can see from the model, this process contains several loops, each of which is executed multiple times, in general.

[3]Available at `http://www.gurobi.com`.
[4]Available at `http://scip.zib.de`.
[5]Available at `http://lpsolve.sourceforge.net`.

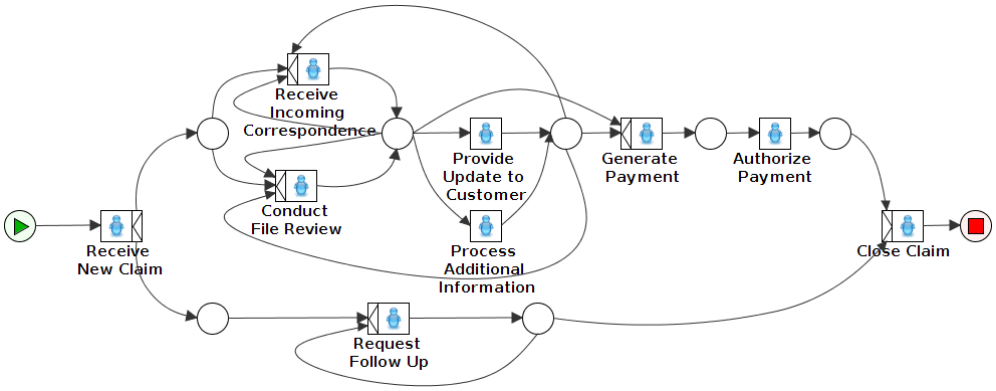


Figure 5.5: The Claims Handling process used for the evaluation.

Four risk analysts working in this insurance company were consulted through an iterative interview process, to identify the risks this process is exposed to.[6] They reported about three equally-important faults related to complete traces σ of the claim handling process:

Over-time fault. This fault is the same as the over-time fault described in Section 5.2. For this risk we set the Maximum Cycle Time $d_{mct} = 30$ (i.e. 30 days) and the maximum duration $d_{\max} = 300$ (i.e. 300 days). The severity of an overtime fault is measured as follows:

$$f_{time}(\sigma) = \max\left(\frac{d_\sigma - d_{mct}}{\max(d_{\max} - d_{mct}, 1)}, 0\right)$$

Customer-dissatisfaction fault. During the execution of the process, if a customer is not updated regularly on their claim, they may feel "unheeded". A customer dissatisfied may generate negative consequences such as negative publicity for the insurance company, leading to bad reputation. In order to avoid this kind of situations, the company's policy is to contact their customers at least once every 15 days. Given the set $\Lambda = \{(t, r, d, \phi) \in \sigma | t = \text{Request Follow Up} \vee t = \text{Receive New Claim} \vee t = \text{Close Claim}\}$ of events belonging to task *Request Follow Up*, to task *Receive New Claim*, or to task *Close Claim*, ordered by timestamp, the severity of this fault is:

$$f_{dissatisfaction}(\sigma) = \sum_{1 \leq i \leq \|\Lambda\|} \max(0, d_{i+1} - d_i - 15days)$$

where d_i is the time stamp of i^{th} event $\in \Lambda$.

Cost Overrun fault. Each task has an execution cost associated with it, e.g.

[6] Three interviews were conducted for a total of four hours of audio recording.

the cost of utilizing a resource to perform a task. Since the profit of the company decreases with a higher number of tasks executed, the company clearly aims to minimize the number of tasks required to process a claim, for example by reducing the number of follow-ups with the claimant or the need for processing additional documents, and reassessing the claim, once the process has started. The severity of the cost overrun fault increases as the cost goes beyond the minimum. Let c_σ be the number of work items executed in σ, $c_{\max}$ be the maximum number of work items (e.g. 30) that should be executed in any process instance that has already been completed (including σ), and $c_{\min}$ be the number of work items with unique label executed in σ. The severity of a cost overrun fault is:

$$f_{cost}(\sigma) = \min\left(\frac{c_\sigma - c_{\min}}{\max(c_{\max} - c_{\min}, 1)}, 1\right)$$

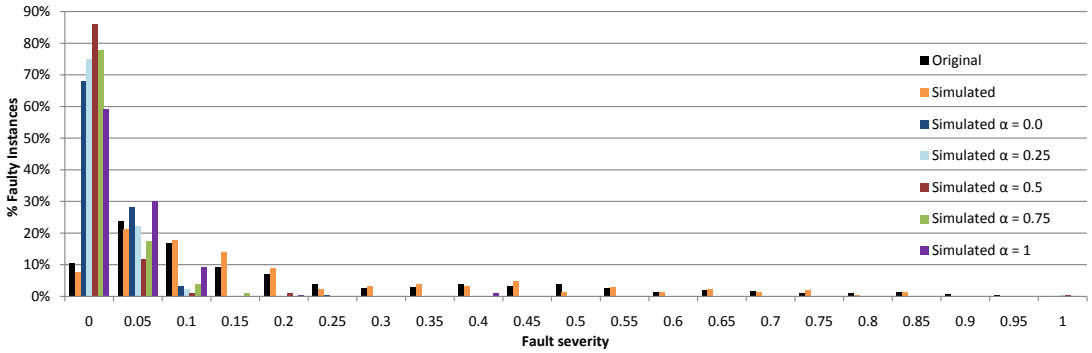
The occurrence of these three faults in the logs is checked using the technique that we proposed in [CLF+13b], which was originally designed for run-time detection of process-related risks.

Trialling our technique within the company was not possible, as the claims handling process concerns thousands of dollars, which cannot be put in danger with experiments. So we had to simulate the execution of this process and the resource behavior using CPN Tools.[7] We mined the control-flow of our simulation model from the original log and refined it with the help of business analysts of the company, and added the data, resource utilization (i.e. who does what), and tasks duration, which we also obtained from the log. We then add the frequency of occurrence of each of these elements, on the basis on that observed from the log. This log was also used to train the function estimators.

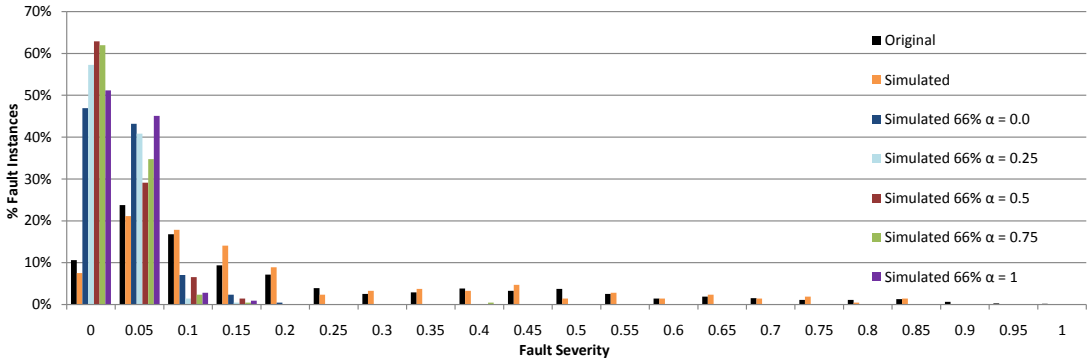
The CPN Tools model we created is a hierarchical model composed of ten nets that all together count 65 transitions and 62 places. The main net is based on the model showed in Figure 5.5, with additional places and transitions in order to guarantee the interaction with our system. The remaining nine nets define the behaviour of each one of the nine tasks showed in Figure 5.5.

We used this model to simulate a constant workload of 50 active instances, in order to maintain a similar ratio to the original log (in the original log we had 271 active instances on average). In order to maintain the ratio between active instances and resources, we reduced the number of resources utilized to one-sixth of the original number observed in the log. Finally, we analyzed the fault

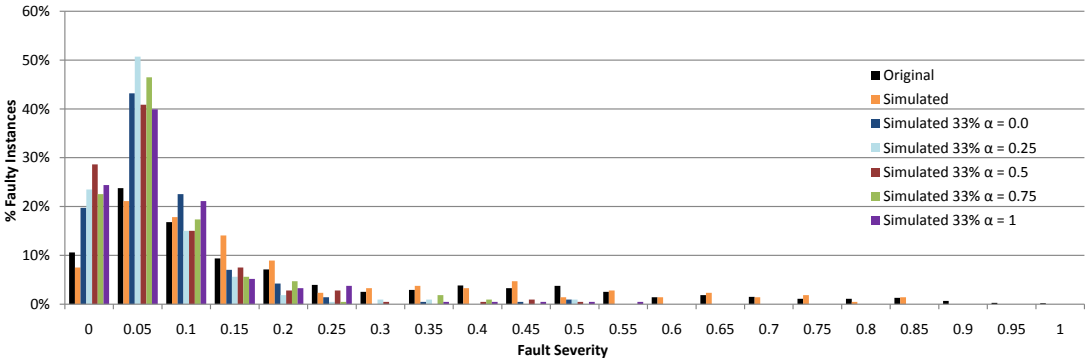
[7] Available at `www.cpntools.org`.



(a) Results following 100% of the suggestions provided.



(b) Results following 66% of the suggestions provided.



(c) Results following 33% of the suggestions provided.

Figure 5.6: Comparison of the fault severity when recommendations are and are not followed, with 0 denoting absence of faults. The x-axis represents the severity of the composite fault and the y-axis represents the percentage of instances that completed with a certain severity.

distribution of the generated log using the technique presented in [CLF+13b].

The model created with CPN Tools was able to reproduce the behavior of the original log. The *Kolmogorov-Smirnov Z* two-samples test (*Kolmogorov-Smirnov* $Z = 0.763$, $p = 0.605 > 0.05$) shows no significant difference between the distribution of the composite fault in the original log and that in the simulated log. This result is confirmed by the *Mann-Whitney* test ($U = 109,163.0$, $z = -0.875$, $p = 0.381 > 0.05$).

Reference Logs	#Traces	% Faulty Instances	Average	Median
Original	1065	89.4%	0.22	0.10
Simulation model	1065	92.5%	0.22	0.15

		Suggestions 100%			Suggestions 66%			Suggestions 33%		
Test Logs	#Traces	% Faulty Instances	Avg	Mdn	% Faulty Instances	Avg	Mdn	% Faulty Instances	Avg	Mdn
Aggregated	1065	26.8%	0.02	0.00	43.9%	0.03	0.00	76.3%	0.07	0.05
- $\alpha = 0.0$	213	31.9%	0.02	0.00	53.1%	0.03	0.05	80.3%	0.08	0.05
- $\alpha = 0.25$	213	24.9%	0.02	0.00	42.7%	0.02	0.05	76.5%	0.07	0.05
- $\alpha = 0.5$	213	14.1%	0.01	0.00	37.1%	0.02	0.05	71.4%	0.07	0.05
- $\alpha = 0.75$	213	22.1%	0.01	0.00	38.0%	0.02	0.05	77.5%	0.07	0.05
- $\alpha = 1.0$	213	40.8%	0.03	0.00	48.8%	0.03	0.05	75.6%	0.08	0.05

Table 5.1: Percentage of faulty instances, mean and median fault severity occurring in the reference logs, i.e. original log and simulation model log. Percentage of faulty instances, mean and median fault severity occurring in the test logs aggregated into a unique log, i.e. simulated aggregated, and for each value of α, reported for each of the three sets of experiments (33%, 66% and 100% suggestions used).

We performed three sets of experiments. In the first set, all the suggestions provided by the system were followed. In the second set, only 66% of the times the suggestions were followed, and executing the process as the company would have done for the remaining 33% of the times. Finally, in the third set of experiments, only 33% of the times the suggestions provided by our system were followed. Moreover, for each set of experiments we tested several values of α (i.e. 0.0, 0.25, 0.5, 0.75 and 1.0), where α equal to 0 will shift focus on reducing risks, while α equal to 1 on reducing the overall execution time (see Section 5.4).

All experiments were executed simulating the execution of the process by means of the CPN Tools model. For each experiment we generated a new log containing 213 fresh log traces (a fifth of the traces contained in the original log). We used a computer with an Intel Core i7 CPU (2.2 GHz), 4GB of RAM, running Lubuntu v13.10 (64bit). We used Gurobi 5.6 as MILP solver as this is the most efficient solver among the three that we support [KAA+11][8] and imposed a time limit of 60 seconds, within which a solution needs to be provided for each problem. For mission-critical processes, the time limit can also be reduced. If a time limit is set and Gurobi cannot find a solution within the limit, a sub-optimal solution is returned, i.e. the best solution found so far. The experiments have shown that, practically, the returned solution is always so close to the optimal that it does not influence the final fault's magnitude.

Figure 5.6 shows the results of each of the three sets of experiments, comparing the fault severity of the original log with that obtained when recommendations are followed. It is worth highlighting how the results are given in terms of severity

[8]Gurobi is free of use for academic purposes but is not open-source. This is the reason why we also support other two implementations: SCIP and LPSolve.

measured for completed instances. Risks are relative to running instances and estimate the expected fault severity and likelihood when such instances complete.

Table 5.1 shows the results of the experiments. In this table we show percentage of faulty instances, mean and median fault severity obtained during our tests. The values are shown for the original log and the log obtained by our simulation model without using our recommendation system (Simulation model). Same values are also reported for each log obtained using our recommendation system, both in an aggregated log (Simulated aggregated) and for each value of α, over the three sets of experiments (33%, 66% and 100% suggestions used). In the best case (Simulated log with $\alpha = 0.5$), our technique was able to reduce the percentage of instances terminating with a fault from 89.4% to 14.1% and the average fault severity from 0.216 to 0.01. In particular, the use of our system significantly reduced the number of instances terminating with faults, as evidenced by the result of the *Person's* χ^2 test ($\chi^2(1) = 857.848$, $p < 0.001$ for the first set of experiments, $\chi^2(1) = 494.907$, $p < 0.001$ for the second set, and $\chi^2(1) = 64.663$, $p < 0.001$ for the third one, computed over the original log and the simulated aggregated log). Based on the *odds ratio*, the odds of an instance completing without a fault are respectively 23.06, 10.75, and 2.62 times higher if our suggestions are followed. Moreover, we tested if the number of suggestions followed influences the effectiveness of our technique. The *Kruskal-Wallis* test ($H(3) = 1,603.61$, $p < 0.001$) shows that the overall fault severity among the three sets of experiments (using the Simulated overall dataset, i.e. independently of the value of the parameter α) and the original log is significantly different, and as revealed by *Jonkheere's* test ($J = 1,658,630.5$, $z = -41.034$, $r = -0.63$, $p < 0.001$), the median fault severity decreases as more suggestions are followed (see Figure 5.7). These two tests indicate that our technique is capable of preventing the occurrence of faults and of reducing their severity. Clearly, it is preferable to follow as many suggestions as possible in order to obtain the best results though this may not always be possible.

We tested how the value of the parameter α influences the effectiveness our technique. We compared the performances obtained with each value of α for each set of experiment. The *Kruskal-Wallis* test ($H(4) = 46.176$, $p < 0.001$ for the first set of experiments, $H(4) = 17.191$, $p = 0.002 < 0.05$ for the second one, $H(4) = 5.558$, $p = 0.235 > 0.05$ for the third one) shows how the value of the parameter α significantly influences the median fault severity if the suggestions proposed are followed in at least 66% of the instances. *Jonkheere's* test ($J = 251,305$, $z = 5.577$, $r = 0.17$, $p < 0.001$ for the first set of experiments, $J = 246,322.5$,

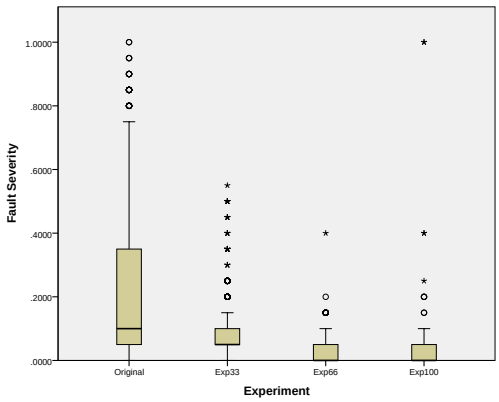


Figure 5.7: BoxPlot showing the fault severity occurring in instances of each of the three experiments and of the original log.

$z = 3.918$, $r = 0.12$, $p < 0.001$ for the second one) revealed that the median fault severity increases when the value of α diverges from 0.5 moving either toward 0 or 1.

In the case study taken in exam, the duration of an instance has an influence over the over-time fault and the cost overrun fault. A short execution time will directly minimize the duration of an instance (thus preventing the over-time fault) but also reduce the number of activities that are executed inside such an instance (thus preventing the cost overrun fault). In light of so, it is not strange that the best results are obtained with $\alpha = 0.5$ which strikes a good balance between minimizing risks and overall execution time.

Finally, we performed a sensitivity test over the time limit granted to the ILP solver. We tested our recommendation system with five different time limits, while keeping the value of α equal to 0.5 and following all suggestions (best configuration for risk prevention). The time limits used were: 5, 10, 20, 40 and 60 seconds. Figure 5.8 shows the distribution of fault severities obtained using these different time limits. We can observe that changing the time limit yields statistically different distributions, as revealed by the *Kruskal-Wallis* test ($H(4) = 74.738$, $p < 0.001$). Moreover, the *Jonkheere's* test ($J = 186,238$, $z = -8.631$, $r = -0.264$, $p < 0.001$) reveals that the median fault severity decreases when more time is granted to the ILP solver. From a practical point of view though, it is interesting to observe that even with a time limit of 5 seconds the approach can still notably reduce the faults severity, with 90% of the instances terminating with a fault severity up to 0.05 out of 1. This suggests that users may set the time limit to be granted to the ILP solver on the basis of the number of process activities that are critical, i.e. using a low time limit if the number of critical activities is low and a high time limit if that number is high.

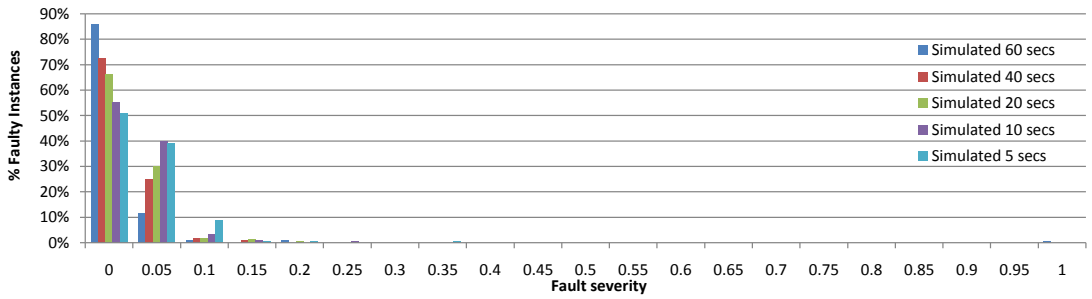


Figure 5.8: Fault severity distribution using different time limits.

Based on the results of our experiments we can conclude that the approach produces a significant reduction in the number of faults and their severity. Specifically, for the case study in question we achieved the best results with α equal to 0.5, with a time limits of 60 seconds. We observe that this parameter can be customized based on the priorities of the company where our approach would be deployed, e.g. an organization may use lower values of α if risk reduction is prioritized over reduction of process duration. The value of α may, for example, be derived from previous process instances. This can be achieved by analysing the logs and measuring the average process duration and its standard deviation. Processes with a high standard deviation will probably indicate that the company is more concerned about the quality of its processes (risk prevention) than its processes' total execution time.

5.7 Related Work

The technique developed in this work can be compared to work in the following areas: job scheduling and work-item distribution.

5.7.1 Job Scheduling

The problem of distributing work items to resources in business process execution shares several similarities with the job-shop scheduling [Ven07, Elm93, Bak74, ZD95]. Job-shop scheduling concerns M jobs that needs to be assigned to a N machines, with $N < M$, while trying to minimize the make-span, i.e. the total length of the schedule. Jobs may have constraints, e.g. job i needs to finish before job j can be started, certain jobs can only be performed by given machines.

Unfortunately, these approaches are intended for different settings and cannot be specialized for risk-informed work-item assignment. To our knowledge, techniques of job-shop scheduling are unaware of the concept of cases or process

instances, since typically jobs are not associated with a case.

The concept of case is crucial when dealing with process-aware information systems. Work items are executed within process instances and many process instances can be running at the same, so like many work items may be enabled for execution. Different instances may be worked on by the same resources and, hence, the allocation within a instances may affect the performance of other instances. Without considering the instances in which work items are executed, an important aspect is not considered and, hence, the overall allocation is not really optimized. Moreover, applying job-scheduling for work-item distribution, such work items will be distributed with a push strategy, i.e. a work item is pushed to a single qualifying resource. This is also related to the fact the jobs are usually assumed to be executed by machines, whereas, in process-aware information systems, work items are normally being executed by human resources. Work items may also be executed by automatic software services, but this is not the situation in the majority of setting. Kumar et al. [KAV02] show that push strategies already perform very poorly when the resource work-load is moderately high. Therefore, work items ought to be distributed with a pull mechanism, i.e. enabled work items are put in a common pool and offered to qualifying resources, which can freely pick any of them. As a matter of fact, a pull strategy is far the most common used in current-day process-aware information systems.

5.7.2 Operational Support

The work proposed in this thesis is also related to body of work that is concerned with devising frameworks and architectures to provide operational support for business processes as a service. For instance, Nakatumba et al. [NWA12] propose a service for operational support which generalizes what is proposed in [SWDA08]. This service is implemented in ProM, a pluggable framework to implement process-aware techniques in a standardized environment. On its own, the service does not implement recommendation algorithms but provides an architecture where such algorithms can be easily plugged in. For instance, the prediction technique in [MDDG14] is an example of algorithm plugged into this architecture (more details on this work are provided in the next subsection). Another example is the work in [MMC+13], which concerns a recommendation algorithm based on monitoring the satisfaction of business constraints. This work does not make any form of prediction nor automatic optimal work-items' distribution.

As a matter of fact, there is no conceptual or technical limitation that would

Table 5.2: Comparison of different approaches for operational support in Process-aware Information Systems

Approach	Weight	Process Perspectives Computation	Optimal Distribution	Objective	Assignment Method
Kim et al. [KOJ14]	Dynamic	Control-flow, Resource	-	Time, Cost	-
Yang [Yan08]	Static	-	Instance level	Customizable	PUSH
Kumar et al. [KDS13]	Dynamic	Control-flow, Resource	Instance level	Cooperation[a]	PUSH
Kumar et al. [KAV02]	Static	-	Instance level	Suitability, Urgency, Workload	PUSH/PULL[b]
Huang et al. [HLD12]	Dynamic	Control-flow, Resource, Data, Time	Instance level	Customizable	PUSH
Van der Aalst et al. [ASS11]	Dynamic	Control-flow	-	Time	-
Folino et al. [FGP12]	Dynamic	Control-flow, Resource, Data, Time	-	Time	-
Van der Spoel et al. [SKA12]	Dynamic	Control-flow	-	Cost	-
Cabanillas et al. [CGR+13]	Static	Control-flow, Resource	Process level	User preference[c]	PUSH
Barba et al. [BWV12]	Static	Control-flow, Resource	Instance level	Time	PULL
Maggi et al. [MDDG14]	Dynamic	Control-flow, Resource, Data	-	Customizable LTL [d] formulas	-
Our technique	Dynamic	Control-flow, Resource, Data, Time	Process Level	Customizable	PULL

[a] Work items are distributed to maximize the quality of the cooperation among resources. This approach assumes that some resources can cooperate better than others when working on a process instance.
[b] Resources declare their interest in picking some work items for performance. The approach assigns each work item to the interested resource that guarantees the better distribution.
[c] At design-time, users provide preferences for work items. At run time, the system allocates work items to resources to maximize such preferences.
[d] The expressiveness power of business goals in the form of a single LTL formula is lower than what our technique allows for. In principle, multiple LTL formulas can be provided though one has to balance contrasting recommendations for the satisfiability of such formulas.

prevent our approach from being implemented as a plug-in for an operational-support service.

5.7.3 Work-item distribution

Our work on work-item distribution to minimize risks shares commonalities with Operational support and Decision Support Systems (DSSs). We aim to provide recommendations to process participants to take risk-informed decisions. Our work fully embraces the aim of these systems to improve decision making within work systems [Alt04], by providing an extension to existing process-aware information systems.

Mainstream commercial and open-source BPM systems do not feature work-item prioritization. They only allow one to indicate a static priority for tasks (e.g. low, medium or high priority), independently of the characteristics of the process instance and of the qualified resources. Similarly, the YAWL system, which is the one we extended, does not provide means for operational support, besides the extension proposed by de Leoni et al. [LAAH12], which, however, defines very basic metrics only.

Several approaches have been proposed in the literature. Table 5.2 summarizes and compares the most significant ones, using different criteria:

Weight Computation. In order to perform an optimal distribution, every work

item needs to be assigned a weight, which may also depend on the resources that is going to perform it or on the moment in time when such work item is performed. These weights can be defined either statically by analysts or dynamically computed on the basis of the past history recorded in an event log.

Process Perspective. When weights are dynamically defined, they may be computed considering different perspectives: control-flow, resources, data and time.

Optimal Assignment. The optimization of work-item distribution can be computed by considering single instances in isolation or trying to optimize the overall performances of all running instances.

Objective. The work-item distribution can be optimized with respect to several factors, such as minimizing the cost, time or maximizing the cooperation. Only few approaches allow one to customize the objective function to minimize/maximize.

Assignment Method. Once an optimal distribution is computed, each work item can be pushed to single qualified resource or, conversely, can be put in a common pool and simply recommended to a single resource.

Among the available approaches only the one by Cabanillas et al. [CGR+13] computes the optimal allocation of resources at the process level. Specifically, this work proposes a priority-based resource allocation, where resources are ranked according to preferences defined using the Semantic Ontology of User Preferences [GRRC10]. Once a work item needs to be executed it is pushed to the resource ranking the highest on the basis of the expressed preferences.

Among the approaches providing optimal distribution only two approaches support a pull assignment. The approach of Barba et al. [BWV12] optimizes process performances, using constraint programming (planning and scheduling problem) where constrains are defined considering control-flow and resources only. On the other hand, the approach of Kumar et al. [KAV02] aims to obtain the right balance between execution time and quality. This approach uses work allocation metrics and various quality attributes to find the optimal allocation strategy keeping into consideration the preference of resources for certain work items.

The approach of Yang [Yan08], similar to all the approaches discussed so far, assigns a static weight to each work item. This approach optimizes process execution time and total execution cost according to user preferences. Preferences are

defined using a multi-attribute utility function that is optimized using the particle swarm optimization algorithm. A second approach by Kumar et al. [KDS13], and the approach of Huang et al. [HLD12], conclude the list of approaches providing optimal distribution of work items. Kumar et al. [KDS13] propose an approach for optimal resource cooperation using integer linear programming to identify the group of resources with the best synergy to perform a process instance while Huang et al. [HLD12] propose to use task operation models.

There are also approaches that focus on prediction only. Van der Aalst et al. [ASS11] propose an approach to predict total execution time and remaining execution time. The approach uses logs to generate transition systems annotated with timing information. Transition systems are employed to provide predictions using similarly completed executions as a reference. Folino et al. [FGP12] use a combination of clustering techniques and transition systems. Using clustering they identify process variants in a log and for each cluster they generate a transition system. When a prediction is required, using decision trees the authors identify which cluster the current instance belongs to, and then use the associated transition system to provide a prediction.

Van der Spoel et al. [SKA12] propose an approach to predict the cash flow of a process. This approach uses a combination of process flow prediction, i.e. predicting how the process execution will proceed, and cost prediction, i.e. predicting how much the execution of a predicted activity will cost. Kim et al. [KOJ14] propose the use of decision trees to minimize completion time or total labor cost, where the resource with the lowest predicted completion time or total labor cost is suggested.

Finally, Maggi et al. [MDDG14] propose a predictive approach to prevent process constraints violation. Users can define linear temporal logic constraints at any point in time during the execution of a process. Then, when a prediction is required, the approach retrieves all traces having a similar prefix of the current instance. These instances are then used to generate a decision tree that is used to predict how the process execution should proceed to satisfy the predefined constraints.

There are also approaches (e.g.,[RMA07, HLD11, LWYS08]) that mine association rules from event logs to define the preferable distribution of work items. However, in the end a resource manager needs to manually assign work items to resources. Manual distributions are clearly inefficient because they are both unlikely to be optimal and some work items probably remain unassigned for a certain amount of time until the manager takes charge of their assignment. More-

over, the mined rules consider process instances in isolation. strongest emphasis is associated with the resource that would minimize the fault's risk).

The last row refers to our technique. This is the only one that performs predictions based on various perspectives and uses such predictions to compute an optimal distribution that is not local to instances but is global at process level. Moreover, we use customizable faults as objective functions. To the best of our knowledge, this is the only approach where each work item is recommended to qualifying resources with different emphasis (the strongest emphasis is associated with the resource that would minimize the fault's risk).

5.8 Summary

This chapter present a recommendation system that allows users to take risk-informed decisions when partaking in multiple process instances running concurrently. Using historical information extracted from process execution logs, for each state of a process instance where input is required from a process participant, the system determines the risk that a fault (or set of faults) will occur if the participant's input is going to be used to carry on the process instance. This input can be in the form of data used to fill out a user form, or in terms of the next work item chosen to be executed.

The system relies on two techniques: one for predicting risks, the other for identifying the best assignment of participants to the work items currently on offer. The objective is to minimize both the overall risk of each process instance (i.e. the combined risk for all faults) and the execution time of all running process instances.

We designed the system in a language-independent manner, using common notions of executable process models such as tasks and work items borrowed from the YAWL language. We then implemented the system as a set of components for the YAWL system. For each user decision, the system provides recommendations to participants in the form of visual aids on top of YAWL models. We also extended the YAWL user form visualizer, to show a risk profile based on the data inserted by the participant for a given form. Although we implemented our ideas in the context of the YAWL system, our recommendation system can easily be integrated with other BPM systems by implementing an interface that allows the communication through the log abstraction layer (see Chapter 4 we showed how it can be integrated with the Oracle BPEL 10g database), and by extending the Map-Based Worklist Handler in order to list work items belonging to a different

BPM system than the YAWL system.

We simulated a real-life process model based on one year of execution logs extracted from a large insurance company, and in collaboration with risk analysts from the company we identified the risks affecting this process. We used these logs to train our system. Then we performed various statistical tests while simulating new process instances following the recommendations provided by our system, and measured the number and severity of the faults upon instance completion. Since in reality it might not always be feasible to follow the recommendations provided, we varied the percentage of recommendations to be followed by the simulated instances. Even when following one recommendation out of three, the system was able to significantly reduce the number and severity of faults. Further, results show that risks can be predicted online, i.e. while business processes are being executed, without impacting on execution performance.

In the next chapter we present the last component of our risk-aware framework. It is a technique for mitigating risk occurring in running business processes through the use of the simulated annealing algorithm, and can be triggered as result of a notification received from the risk monitoring technique presented in Chapter 4.

Chapter 6

Risk Mitigation

The two techniques presented so far, i.e. a technique to model risks in executable business process models and detect them as early as possible during process execution, and a technique to provide suggestions during process execution with the intent of preventing the eventuation of risks, leave a final shortcoming affecting approaches for risk-aware business process management. It is the lack of support for automated risk mitigation.

Preventing the eventuation of a risk, or detecting it in time is often not enough to avoid the negative outcome associated with it. A *prompt* risk mitigation should be taken to restore the process instance to a safe state, before the instance progresses any further. Moreover, taking the *right* mitigation at the right time may make the difference between success and failure. In fact, the number of possible ways a process-related risk may be mitigated is potentially very large that it is difficult for a process administrator to take the right decision at the right time, without any support. One has to consider all mitigations that are possible, given the current state of the process instance (including a snapshot of the associated data and resources), and the context in which the instance is running, i.e. the state of other running instances, to make such a decision. For example, in order to mitigate the risk of a process instance A to run overtime, a mitigation may entail to reallocate resources from a process instance B (potentially of another process) to A.

In light of the above, in this chapter we present a technique for automatically mitigating process-related risks, that will complete our risk-aware framework. Since a process instance may be affected by multiple risks at the same time, we treat this problem as a *multi-objective optimization problem*. A solution to this problem is a variant of the risky process instance obtained by applying a sequence of mitigation actions, in order to reduce the risks' probability down to a tolerable

level, or in the best case, to annul the risks altogether. Mitigation actions include control-flow aspects (e.g. skipping a task to be executed), process resources (e.g. reallocating a resource to a different task), and data (e.g. rolling back an executed task to restore its input data). To explore the potentially large solution space, we use dominance-based Multi-Objective Simulated Annealing (MOSA) [SEF+08]. At each run, the algorithm generates a small set of solutions similar to the original process instance but with less risks. It stops when either a maximum number of *non-redundant* solutions (i.e. solutions proposing different mitigations) is found or a given time-frame elapses. This approach is not meant to replace human judgment. Instead, it aims to support process administrators in deciding what mitigations to take, by reducing the number of feasible options, and consequently the time needed to take a decision.

We defined the mitigation actions in collaboration with an Australian risk consultant. To prove the feasibility of this approach, we implemented these actions and the MOSA algorithm on top of the YAWL system. We instantiated a set of process models from the logistics [Vol11] and screen business [OLH+08] domains that are affected by one or more risks, and executed a series of tests to mitigate such risks. The tests show that the technique can find a set of possible solutions within a few minutes of computation, and that in all cases the associated risks are mitigated.

The rest of this chapter is organized as follows. Section 6.1 introduces the running example. Section 6.2 and Section 6.3 introduce preliminary concepts while Section 6.4 describes the proposed technique to mitigate process risks. Section 6.5 illustrates the implementation of the technique, which is then evaluated in Section 6.6. Section 6.7 covers related work and Section 6.8 concludes the chapter.

The research presented in this chapter has been the subject of two publications [CHLA12, CLH+13].

6.1 Running example

Let us now consider an example process for which we have defined several risks, to understand how risk conditions can be formulated in terms of process model elements. These conditions will provide input for the risk mitigation technique presented in the next section. The example process, shown in Figure 6.1, describes the *Payment* subprocess of an order fulfillment process, that we already described in Chapter 4.

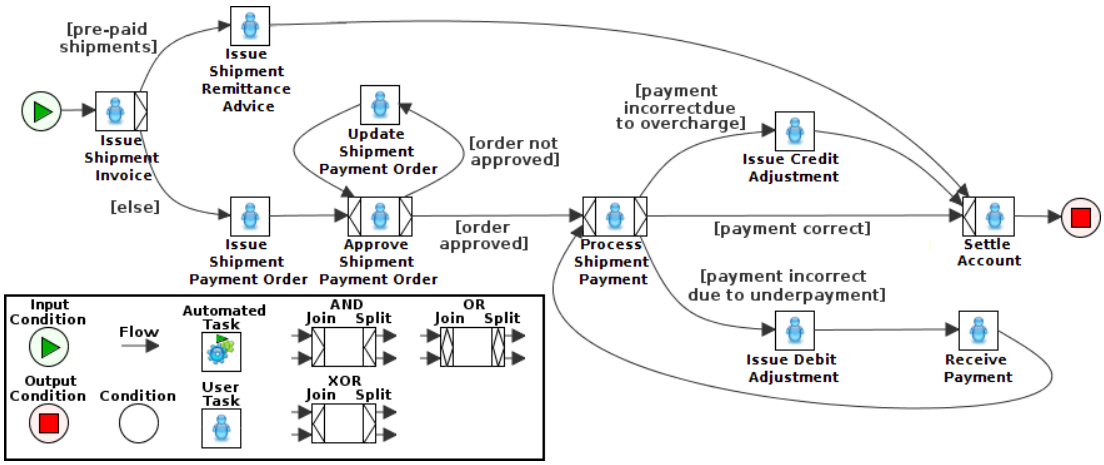


Figure 6.1: Order Fulfillment: Payment subprocess.

In this process, as already illustrated in Chapter 4, we can identify various faults that may occur during execution. For example, a Service Level Agreement (SLA) may establish that the process (or one of its tasks) may not last longer than a Maximum Cycle Time MCT (e.g. 5 days), otherwise a pecuniary penalty may be incurred. To detect the risk of *overtime fault* at run-time, we should check the likelihood that the running instance does not exceed the MCT based on the amount of time T_c expired to that point. Let us consider T_e as the remaining cycle time, i.e. the amount of time estimated to complete the current instance given T_c based on past executions, which can be computed using the approach in [ASS11]. Then the probability of exceeding MCT can be computed as $1 - MCT/(T_e + T_c)$ if $T_e + T_c > MCT$ and is equal to 0 if $T_e + T_c \leq MCT$. If this probability is greater than a tolerance value (e.g. 60%), we notify the risk to the user.

A second fault is related to the resources participating in the process. The Senior Finance Officer who has approved a Shipment Payment Order for a given customer must have not approved another order by the same customer in the last d days, otherwise there is a potential for *approval fraud*, a violation of a four-eyes principle across different instances of the Payment subprocess. To detect this risk we first have to check that there is an order, say order o of customer c, to be approved. Moreover, we need to check that either of the following conditions holds: i) o has been allocated to a Senior Finance Officer who has already approved another order for customer c in the last d days; or ii) at least one Senior Finance Officer is available who approved an order for customer c in the last d days and all other Senior Finance Officers who did not approve an order for c during the last d days are unavailable.

Finally, a third fault relates to a situation where a process instance executes

a given task too many times, typically via a loop. Not only could this lead to a process slowdown, but also to a "livelock" if the task is in a loop whose exit condition is deliberately never met. In general, given a task t, a maximum number of allowable executions of t per process instance, $MAE(t)$, can be fixed as part of the service-level agreement for t. In our example, this fault may occur if task "Update Shipment Payment Order" is re-executed five times within the same process instance. We call this an *order unfulfillment* fault. To detect the risk at run-time, we need to check if: i) the Update task is currently being performed for order o; and ii) it is likely that the task will be repeated within the same process instance. The probability that the number of times a task will be repeated within the same instance is computed by dividing the number of instances where the MAE for the task has been reached by the number of instances that have executed this task at least as many times as it has been executed by the current instance, and have completed. If the probability to exceed $MAE(t)$ is greater than a tolerance value for t, e.g. 60%, we notify the risk to the user.

In the next section we present a set of mitigation actions that can be performed "on-the-fly" on a running process instance in order to mitigate its risks.

6.2 Preliminaries

Before presenting our technique for automated risk mitigation we introduce a number of preliminary concepts and notations. We will not repeat the definition of a YAWL specification presented in Chapter 3, but we will link some additional concepts to it. During execution, each task belonging to a YAWL specification that needs to be executed produce the instantiation of a work item. These work items may go through various statuses. The set *StatusType* contains the various statuses that a work item may go through during its lifecycle. These are: *offered*, *allocated*, *started*, *completed*, *forceCompleted*, *cancelled*, *failed*, *deadlocked* used by the YAWL system and additionally *deoffered*, *deallocated*, *destarted*, *rollback*, *skipped* used for mitigation purposes. Many of these statuses are self-explanatory. The status *rollback* is the status of a work item which was completed but then enabled again though not *offered*. The status *skipped* is the status of a work item that was skipped, which is similar to the status *completed* but the work item was not actually performed. For convenience, we provide certain groupings of event types. In particular, $Rel \triangleq StatusType \setminus \{cancelled, failed, rollback\}$ is the set of event types that identify a work item as subject to mitigation. $Active \triangleq \{offered, allocated, started\}$ is the set of event types that mark a work

item as in progress, $Completed \triangleq \{completed, forceCompleted\}$ is the set of event types that mark a work item as completed, and $ActiveC : Active \cup Completed$ is their union.

Given set $ActiveC$ we define a partial order $\preccurlyeq \subseteq ActiveC \times ActiveC$ such that it preserves the partial ordering $deoffered < offered < allocated < started < completed = forceCompleted$.

6.3 Event and Work Item Comparison

When we consider a YAWL Log, i.e. a log resulting from the execution of a YAWL specification, temporal ordering relations between events associated to work items are not immediately visible. In order to cope with so we are introducing some concepts that will simplify the temporal comparison between events.

Definition 8 (Event Comparison)**.** *Let $L = (\mathcal{E}, \mathcal{W}, \mathcal{C}, Model, WI, Case, Task, EvType, Time, Res, Inp, Outp)$ be a log, given $\mathcal{E}' \subseteq \mathcal{E}$, $\mathcal{E}' \neq \varnothing$, we define the operators $e_1 < e_2$ iff $Time(e_1) < Time(e_2)$ and $e_1 \leqslant e_2$ iff $Time(e_1) \leq Time(e_2)$, which reflect the temporal ordering on events, and the operators $min\ \mathcal{E}' = e_1$ iff $e_1 \in \mathcal{E}'$ and for all $e_2 \in \mathcal{E}', e_1 \leqslant e_2$, which determines the earliest event of an event set, and $max\ \mathcal{E}' = e_1$ iff $e_1 \in \mathcal{E}'$ and for all $e_2 \in \mathcal{E}', e_2 \leqslant e_1$, which determines the latest event of an event set.*

Useful is the possibility of identifying events belonging to the same work item.

Definition 9 (Work Item Event Grouping)**.** *Let L be a log, e an event in this log, $e \in \mathcal{E}$, and w a work item in this log, $w \in \mathcal{W}$, we define the set of events that belong to work item w as $\overline{WI}(w) \triangleq \{e \in E \mid WI(e) = w\}$. Similarly, we define the set of events that belong to the same work item of e as $\overline{WI}(e) \triangleq \overline{WI}(WI(e))$. Finally, the latest event for work item w is defined as $\omega_w \triangleq max\ \overline{WI}(w)$.*

As for events we are interested in being able to compare work items.

Definition 10 (Work Item Comparison)**.** *Let L be a log, with $w_1, w_2 \in \mathcal{W}$, we define $w_1 < w_2$ as $max\ \overline{WI}(w_1) < min\ \overline{WI}(w_2)$. This operator reflects the partial temporal order between work items, i.e. work item w_1 precedes work item w_2 if its latest event is earlier than the earliest event of w_2.*

6.4 Process Risk Simulated Annealing

In this chapter we deal with the problem of automatically mitigating one or more business process risks for a specific running process instance (*case* for short),

Algorithm 3: PRSA Algorithm

Data: Case C, MitigationGraph G, Mitigation M, InitialTemperature initemp, NumberIterations n, NumberTemperatureDrops k
Result: Set of Mitigations F
begin
  $F \Leftarrow \{(G, M)\}$;
  for $i \Leftarrow 1$ **to** n **do**
    $t \Leftarrow initemp$;
    $(G, M) \Leftarrow Any(F)$;
    for $j \Leftarrow 1$ **to** k **do**
      $(G_1, M_1) \Leftarrow applyRandomMitigation(C, G, M)$;
      $\tilde{F} \Leftarrow F \cup \{(G, M)\} \cup \{(G_1, M_1)\}$;
      $\tilde{F}_1 \Leftarrow \{(G_x, M_x) \in \tilde{F} | (G_x, M_x) \prec (G_1, M_1)\}$;
      $\tilde{F}_2 \Leftarrow \{(G_x, M_x) \in \tilde{F} | (G_x, M_x) \prec (G, M)\}$;
      $\delta E \Leftarrow \frac{1}{|\tilde{F}|} \left(\left| \tilde{F}_1 \right| - \left| \tilde{F}_2 \right| \right)$;
      $u \Leftarrow rand(0, 1)$;
      if $u < \min(1, \exp(-\delta E/t))$ **then**
        $(G, M) \Leftarrow (G_1, M_1)$;
        if $(G_x, M_x) \not\prec (G, M)\ \forall (G_x, M_x) \in F$ **then**
          $F \Leftarrow \{(G_x, M_x) \in F | (G, M) \not\prec (G_x, M_x)\} \cup \{(G, M)\}$;
        end
      end
      $t \Leftarrow updateTemperature(t)$;
    end
  end
  return F
end

without raising other business process risks for the same case. This problem belongs to the family of multi-objective optimization problems, and we propose the use of simulated annealing for finding a Pareto-optimal solution, or a set of such solutions. The advantage of using simulated annealing compared to other optimization techniques, such as integer linear programming or genetic algorithms, is that it can provide a sufficiently good solution in a limited amount of time starting with a single element, i.e. the instance that needs to be mitigated.

The Process Risk Simulated Annealing (PRSA) algorithm (see Algorithm 3) is an application of the DBMOSA [SEF+08] algorithm where at each iteration a new solution is discovered through the use of one or more random mitigation actions. The algorithm proposes a solution, or *mitigation*, as a sequence of elementary mitigation actions. A "behavioral cost" (cost for short) is associated with each mitigation action, and measures how deeply an action affects the process instance

to which it is applied. For example, allocating a different resource to a work item has a lower cost than skipping a task that has to be executed. The total cost of a solution is the sum of the costs of each mitigation action used in that solution. A good solution to the PRSA algorithm is one that reduces the likelihood of a risk under its threshold, keeping the total cost as low as possible.

When comparing solutions that have the same cost, a solution that fully mitigates a risk is better than one that mitigates that risk because its risk condition is no longer evaluable. And in turn, this solution is better than one that does not mitigate the risk at all. Finally, if two solutions mitigate the same risk, we privilege the one that yields the lowest risk probability. Given two solutions a, b we say that a dominates b if it mitigates the same risks mitigated by b with a lower total cost. As result, we define them as mutually non-dominating if neither one dominates the other.

Below we describe the more elementary mitigation actions that can be used to create a solution, and how they affect a process case. We use the YAWL language as a reference language to define the mitigation actions, since this language has a formal foundation on which we can build our definitions and algorithms. However, the notions presented in this section can easily be generalized to other languages.

6.4.1 Execution and Mitigation Graph

The concepts previously defined allows us to organize work items in a logical structure. This structure is unable to represent logical relations like parallelism, for this reason we will introduce the concept of execution graph. An *execution graph* for a process case provides a view of its execution and is defined on the basis of a log and its corresponding process model.

Definition 11 (Execution Graph)**.** *Let L be a process log with case c, Y its YAWL specification, and UserID the set of resources, we define the execution graph of c as $G(c) = (Node, TaskN, Status, \rightsquigarrow, ResN, TimeN, VarN)$ where:*

- $Node = \{w \in \mathcal{W} \mid EvType(\omega_w) \in Rel \wedge Case(w) = c\}$ *is the set of nodes, where each node represents a work item that is not modifiable,*
- $TaskN = Task_{|Node}$ *is the restriction of the function Task to the set of nodes,*
- $Status = \{(\omega_w, s) \in Node \times Rel \mid s = EvType(\omega_w)\}$ *is a function relating a node with its status of execution,*

- $\rightsquigarrow = \{(w_1, w_2) \in Node \times Node \mid Status(w_1) \in \{completed, skip\} \wedge TaskN(w_1) \in \circ TaskN(w_2) \wedge \nexists w_3 \in Node[(TaskN(w_1) = TaskN(w_3) \vee TaskN(w_2) = TaskN(w_3)) \wedge w_1 < w_3 \wedge w_3 < w_2])\}$ *is the flow relation between work items. Its reflexive transitive closure is defined as* $\rightsquigarrow^*$,
- $ResN = \{((w, s), r) \in (Node \times Active) \times 2^{UserID} \mid \exists e_1 \in \overline{WI}(w)[EvType(e_1) = s \wedge r = Res(e_1) \wedge \nexists e_2 \in \overline{WI}(w)[e_1 < e_2 \wedge EvType(e_2) \preccurlyeq s]]\}$ *is a function that yields the resources that are involved in the latest changing* w *to status* s,
- $TimeN = \{((w, s), t) \in (Node \times ActiveC) \times \mathcal{T}) \mid \exists e_1 \in \overline{WI}(w)[EvType(e_1) = s \wedge t = Time(e_1) \wedge \nexists e_2 \in \overline{WI}(w)[e_1 < e_2 \wedge EvType(e_2) \preccurlyeq s]]\}$ *is a partial function that yields the timestamp when* w *latest moved to status* s,
- $VarN = \{((w, x), v) \in (Node \times Var) \times \Omega \mid EvType(\omega_w) \notin \{skip, deoffered\} \wedge v = Inp(max\ \{e_2 \in \overline{WI}(w) \mid EvType(e_2) = offered\}, x)\} \oplus \{((w, x), v) \in (Node \times Var) \times \Omega \mid EvType(\omega_w) \in Completed \wedge v = Outp(\omega_w, x)\}$ *is a partial function relating nodes and variables to values.*

As we explore mitigation options the execution graph should evolve along with it, and the initial execution graph becomes a dynamic data structure from which we can modify nodes. We will refer to this modified execution graph as *mitigation graph.*

The concept of *border* identifies work items that can be modified. Such work items are currently in execution, or they are completed work items for which there are no successor work items that are completed or being executed.

Definition 12 (Border)**.** *Let* G *be a mitigation graph. We define the border of* G, $\circlearrowright_G$, *as* $\{n_1 \in Node \mid \forall n_2 \in Node[n_1 \rightsquigarrow^* n_2 \Rightarrow Status(n_2) \in \{deoffered, skipped, rollback\}]\}$.

Discovering which work items are related to a specific resource can be relevant. For this reason we introduce the function $\circledast$ which takes as argument an active event type and a resource, and yields the active border work items associated with that resource.

Definition 13 (Resource Involvement)**.** *Let* L *be a log,* r *a resource,* $r \in UserID$, *and* s *an active event type,* $s \in Active$, *then* $\circledast(r, s) \triangleq \{w \in \mathcal{W} \mid \exists e \in \mathcal{E}[e = \omega_w \wedge EvType(e) = s \wedge r \in Res(e)]\}$ *We will abbreviate active event types to their first letter, e.g. we will write* $\circledast(r, o)$ *instead of* $\circledast(r, offered)$.

To reassign a resource we need to know if the resource is associated with process cases which risks could be eventuate. To obtain an answer to this question

we define the concept of *safe resource collection*. Note that in practice we need to know the process model and the resource model in order to obtain this collection.

Definition 14 (Safe Resource Collection)**.** *Let L be a log, UserID the set of resources, and $\mathcal{C}$ the set of cases, the safe resource collection is defined as $SRC = (PI, Risk)$ where:*

- *$PI : UserID \rightarrow 2^{\mathcal{C}}$ is a function relating resources to sets of active cases, in particular given a resource r returns the set of active cases for which there is a task associated to a role, which resource r belongs to,*
- *$Risk : \mathcal{C} \rightarrow \{risky, norisky\}$ is a function that tells us whether a certain case is considered risky or not.*

Definition 15 (Mitigations)**.** *Let Y be a set of YAWL specifications, with associated set of tasks T_Y, a set of resources UserID, and a log L. A mitigation is represented as $M = (\mathcal{A}, TypeA, TaskA, ResA, CaseA, \succ)$ where:*

- *$\mathcal{A}$ is a set of mitigation actions,*
- *$TypeA : \mathcal{A} \rightarrow \{deoffer, deallocate, destart, offer, allocate, start, rollback, skip\}$, is a function relating actions to types of mitigation,*
- *$TaskA : \mathcal{A} \rightarrow T_Y$ is a function relating actions to tasks,*
- *$ResA : \mathcal{A} \rightarrowtail UserID$ is a partial function relating actions to resources,*
- *$CaseA : \mathcal{A} \rightarrow \mathcal{C}$ is a function relating actions to cases,*
- *$\succ \subseteq \mathcal{A} \times \mathcal{A}$ is a total ordering on mitigation actions indicating the order in which they need to be performed. We refer to this total ordering as the* mitigation sequence*.*

The insertion of a new mitigation action $a \notin \mathcal{A}$ into mitigation M, can be expressed as:

$$addMit(M, a, et, t, r, c) \triangleq (\mathcal{A} \cup \{a\}, TypeA \cup \{(a, et)\}, TaskA \cup \{(a, t)\}, ResA \cup \{(a, r)\}, CaseA \cup \{(a, c)\}, \succ \cup \{(x, a) \mid x \in \mathcal{A}\})$$

6.4.2 Mitigation Actions

Now we are in a position to introduce the mitigation actions. For each action we will provide a short description of its behavior; we will quantify its cost and specify the precondition(s) required for its application. All these actions are executed in the context of a mitigation M. As soon as a risk is detected we collect the log L containing all process cases. This log is used to generate an

execution graph G', that we refer to as the original execution graph. It is used as a reference for comparison with the original status of the system. The effects of mitigations actions are explored, though not yet applied, during execution of the mitigation algorithm, and hence they are performed on a clone of the original execution graph which we will refer to as G.

Throughout the remainder of this section G' is the original execution graph, G the mitigation graph in use, and $c \in C$ is a case. Moreover, whenever a node is modified, we need to store the time this modification occurred. In order to capture the time, we use function $curr()$.

A mitigation is a sequence of mitigation actions. Below we describe the mitigation actions supported by the PRSA algorithm, and the effects that each action yields.

Algorithm 4: Deoffer Task

function *deOff (Case c, Mitigation Graph G, Mitigation M)*;
Output: Mitigation Graph G, Mitigation M
begin
  $n \Leftarrow Any(\{x \in \circlearrowleft_G \mid Status(x) = offered\})$;
  if $n \neq \perp$ **then**
    $r \Leftarrow Any(ResN(n, offered))$;
    if $|ResN(n, offered)| > 1$ **then**
      $et \Leftarrow offered$;
      $TimeN \Leftarrow TimeN \oplus \{((n, offered), curr())\}$;
      $ResN \Leftarrow ResN \oplus \{((n, offered), ResN(n, offered) \setminus \{r\})\}$;
    else
      $et \Leftarrow deoffered$;
      $TimeN \Leftarrow \{(n, offered)\} \triangleleft TimeN$;
      $ResN \Leftarrow ResN(n, offered) \triangleleft ResN$;
      $VarN \Leftarrow \{(n, v) \mid VarN(n, v) \in \Omega\} \triangleleft TimeN$;
    $Status \Leftarrow Status \oplus \{(n, et)\}$;
    $M \Leftarrow addMit(M, NewAction(), deoffer, TaskN(n), r, c)$;
  return (G, M)

Deoffer This action deoffers a task from a resource to whom the task was offered. We can execute $deOff(c, G, M)$ as described in Algorithm 4 if there is a work item $x \in \circlearrowleft_G$ such that x is an *offered* work item. The cost of this action was set to 1 and this action serves as a reference for the cost of the other actions. With reference to our working example, let us assume that for certain process instances an order cannot be updated (e.g. when an order's line items have already gone into production their quantity can no longer be reduced). To prevent such update, we can set a risk condition that is satisfied as soon as a

Algorithm 5: Deallocate Task

function *deAll(Case c, Mitigation Graph G, Mitigation M)*;
Output: Mitigation Graph G, Mitigation M
begin
  $n \Leftarrow Any(\{x \in \circlearrowleft_G \mid Status(x) = allocated\})$;
  if $n \neq \perp$ **then**
    $r \Leftarrow Any(ResN_G(n, allocated))$;
    $TimeN_G \Leftarrow \{(n, allocated)\} \triangleleft TimeN_G$;
    $Status_G \Leftarrow Status_G \oplus \{(n, offered)\}$;
    $ResN_G \Leftarrow ResN_G \oplus \{((n, allocated), \varnothing)\}$;
    $M \Leftarrow addMit(M, NewAction(), deallocate, TaskN_G(n), r, c)$;
  return (G, M)

work item of task "Update Shipment Payment Order" is offered to a resource in a specific instance. This risk can be annulled by deoffering this work item and then skipping the work item altogether to prevent it from being reoffered.

Deallocate This action deallocates a task from the resource to whom the task was allocated. If there is a work item $x \in \circlearrowleft_G$ such that x is an *allocated* work item, we can execute $deAll(c, G, M)$ (see Algorithm 5). We set the cost of this action to 2, since considering the progress status of a work item, deallocating a work item should be more "expensive" than deoffering it. In the Payment subprocess this action could be used to mitigate the approval fraud risk. The work item of "Approve Shipment Payment Order" can be deallocated from the resource to whom this work item is allocated when the risk is detected, since this resource approved another order for the same customer in the past.

Destart This action brings an already started work item back to the state *allocated* and allocates it to the resource who started it. We can execute the action $deSta(c, G, M)$, as described in Algorithm 6, if there is a work item $x \in \circlearrowleft_G$ such

Algorithm 6: Destart Task

function *deSta(Case c, Mitigation Graph G, Mitigation M)*;
Output: Mitigation Graph G, Mitigation M
begin
  $n \Leftarrow Any(\{x \in \circlearrowleft_G \mid Status(x) = started\})$;
  if $n \neq \perp$ **then**
    $r \Leftarrow Any(ResN_G(n, started))$;
    $TimeN_G \Leftarrow \{(n, started)\} \triangleleft TimeN_G$;
    $Status_G \Leftarrow Status_G \oplus \{(n, allocated)\}$;
    $ResN_G \Leftarrow ResN_G \oplus \{((n, started), \varnothing)\}$;
    $M \Leftarrow addMit(M, NewAction(), destart, TaskN_G(n), r, c)$;
  return (G, M)

Algorithm 7: Offer Task

function *off (Distribution Set D, Case c, Mitigation Graph G, Mitigation Graph G′, Mitigation M)*;
Output: Mitigation Graph G, Mitigation M
begin
  $n \Leftarrow Any(\{x \in \circlearrowleft_G \mid Status_G(x) \in \{offered, deoffered\} \wedge Status_{G'}(x) \in Active\})$;
  if $n \neq \perp$ **then**
    $r \Leftarrow Any(D(TaskN(n)) \setminus ResN_G(n, offered) \cup ResN_{G'}(n, offered))$;
    if $r \neq \perp$ **then**
      $TimeN_G \Leftarrow TimeN_G \oplus \{((n, offered), curr())\}$;
      $VarN_G \Leftarrow TimeN_G \oplus \{VarN_{G'}(n)\}$;
      $Status_G \Leftarrow Status_G \oplus \{(n, offered)\}$;
      $ResN_G \Leftarrow ResN_G \oplus \{((n, offered), ResN_G(n, offered) \cup \{r\})\}$;
      $M \Leftarrow addMit(M, NewAction(), offer, TaskN_G(n), r, c)$;
  return (G, M)

that x is a *started* work item. For this action we set the cost to 3 as destarting a work item requires more effort than deallocating a work item. The *destart* action may also be used to mitigate an approval fraud risk. For example, it may be used to "free up" a resource who has never approved an order for the current customer, reducing this way the probability of allocating the work item of "Approve Shipment Payment Order" to a resource who has approved another order for the same customer in the past.

Offer This action (see Algorithm 7) offers a work item to a resource to whom the task is not currently offered, either because it is not yet part of the set of resources to whom the task is currently offered, or because the task is currently *deoffered*. Given a function D that relates tasks to the set of resources to whom their work items can be offered, we can execute $off\,(D, c, G, G', M)$ if there is a work item $x \in \circlearrowleft_G$ such that x is an *offered* or *deoffered* work item, and this work item is an *offered*, *allocated* or *started* work item in the execution graph G'. This action has a cost of 1, the same as *deoffer*. Since this action can only be executed if we previously executed a *deoffer*, these two actions can be combined to "reoffer" a work item to another resource (with a total cost of 2). For example, to reduce the risk of approval fraud in the Payment subprocess, we can deoffer the work item of "Approve Shipment Payment Order" from all Senior Financial Officers that have already approved another order for the same customer in the past, and offer that work item to a Senior Financial Officer that did not.

Allocate This action reallocates a work item that was deallocated before (and still has not been allocated) to a resource to whom the task was not allocated when

Algorithm 8: Allocate Task

function *all(Case c, Mitigation Graph G, Mitigation Graph G′, Mitigation M)*;
Output: Mitigation Graph G, Mitigation M
begin
  $n \Leftarrow Any(\{x \in \circlearrowleft_G \mid Status_G(x) = offered \wedge Status_{G'}(x) \in \{allocated, started\}\})$;
  if $n \neq \perp$ **then**
    $r \Leftarrow Any(ResN_G(n, offered) \setminus \{ResN_{G'}(n, allocated)\})$;
    if $r \neq \perp$ **then**
      $TimeN_G \Leftarrow TimeN_G \oplus \{((n, allocated), curr())\}$;
      $Status_G \Leftarrow Status_G \oplus \{(n, allocated)\}$;
      $ResN_G \Leftarrow ResN_G \oplus \{((n, allocated), r)\}$;
      $M \Leftarrow addMit(M, NewAction(), allocate, TaskN_G(n), r, c)$;
  return (G, M)

the deallocation took place (see Algorithm 8). We can execute $all(c, G, G', M)$ if there is a work item $x \in \circlearrowleft_G$ such that x is an offered work item, and x is originally an *allocated* or *started* work item. This action has a cost of -1. This action can only be executed if we previously executed a *deallocate*, with the result of changing the resource involved in a work item (so the total cost is $2 - 1 = 1$). We can use the combination *deallocate* + *allocate* as an alternative to mitigate the risk of approval fraud. With a total cost of 1, this combination would be preferred to using *deoffer* + *offer*.

Algorithm 9: Start Task

function *sta(Case c, Mitigation Graph G, Mitigation Graph G′, Mitigation M)*;
Output: Mitigation Graph G, Mitigation M
begin
  $n \Leftarrow Any(\{x \in \circlearrowleft_G \mid Status_G(x) = allocated \wedge Status_{G'}(x) = started\})$;
  if $n \neq \perp$ **then**
    $r \Leftarrow \{ResN_G(n, allocated)\} \setminus \{ResN_{G'}(n, started)\}$;
    if $r \neq \perp$ **then**
      $TimeN_G \Leftarrow TimeN_G \oplus \{((n, started), curr())\}$;
      $Status_G \Leftarrow Status_G \oplus \{(n, started)\}$;
      $ResN_G \Leftarrow ResN_G \oplus \{((n, started), r)\}$;
      $M \Leftarrow addMit(M, NewAction(), start, TaskN_G(n), r, c)$;
  return (G, M)

Start This action (see Algorithm 9) restarts a work item that was previously destarted (and has not yet been restarted) and associates it with a different

Algorithm 10: Rollback Task

function *rollbackTask(Case c, Mitigation Graph G, Mitigation M)*;
Output: Mitigation Graph G, Mitigation M
begin
  $n_1 \Leftarrow Any(\{x \in \circlearrowleft_G \mid Status_G(x) \in Completed\})$;
  if $n_1 \neq \perp$ **then**
    foreach $n_2 \in Node_G$ **do**
      if $n_1 \rightsquigarrow^*_G n_2$ **then** $Status_G \Leftarrow Status_G \oplus \{(n_2, rollback)\}$;
    $Status_G \Leftarrow Status_G \oplus \{(n_1, rollback)\}$;
    $M \Leftarrow addMit(M, NewAction(), rollback, TaskN_G(n_1), \perp, c)$;
  return (G, M)

resource from the one who started the task. We can execute $sta(c, G, G', M)$ if there is a work item $x \in \circlearrowleft_G$ such that x is an *allocated* work item, and x is originally a *started* work item. The cost of this action is -1, and the reasoning is similar to that used for the *allocate* action. This mitigation action can be used to reduce the negative impact of a *deoffer* or *deallocate* previously performed on a process instance.

Algorithm 11: Skip Task

function *skip(Case c, Mitigation Graph G, Mitigation M, YAWL Specification Y)*;
Output: Mitigation Graph G, Mitigation M
begin
  $D \Leftarrow \{t \in T_{Model(c)} \mid \exists x \in \circlearrowleft_G [Status_G(x) = deoffered \wedge TaskN_G(x) = t]\}$;
  $R \Leftarrow \{t \in T_{Model(c)} \mid \exists x \in \circlearrowleft_G [Status_G(x) = rollback \wedge TaskN_G(x) = t]\}$;
  $U \Leftarrow \{t \in T_{Model(c)} \mid \exists x \in \circlearrowleft_G [t \in TaskN_G(x) \circ^*]\}$;
  $t \Leftarrow Any((D \cup R) \cap U)$;
  $a \Leftarrow \exists a \in \mathcal{A} \mid TypeA(a) = skip \wedge TaskA(a) = t \wedge CaseA(a) = c$;
  if $t \neq \perp \wedge skippable(t) = yes \wedge \neg a$ **then**
    $M \Leftarrow addMit(M, NewAction(), skip, t, \perp, c)$;
  return (G, M)

Rollback This action returns a completed work item to the status of unoffered. We can execute $rollbackTask(c, G)$ if there is a work item $x \in \circlearrowleft_G$ such that x is a *completed* work item. Its operationalization is described in Algorithm 10. The rollback action restores the case to a consistent status where the execution of a given work item never happened. A compensation routine can be associated with a task, so that it is triggered when the task is rolled back. The idea of this compensation routine is to deal with elements outside the control of the workflow engine (e.g. returning the money to a client after their payment has been rolled

Algorithm 12: Relocate Resource

function *relRes(Case c, Mitigation Graph G, Mitigation M, Safe Resource Collection SRC)*;
Output: Mitigation Graph G, Mitigation M
begin
  $n \Leftarrow Any(\{x \in \circlearrowright_G \mid Status(x) = started\})$;
  $r_1 \Leftarrow ResN_G(n, started)$;
  $t \Leftarrow \perp$;
  foreach $r_2 \in ResN_G(n, offered)$ **do**
    if $t = \perp \wedge |PI(r)| = 1 \wedge Risk(Any(PI(r))) = norisky \wedge (|\circledast(r,s)| + |\circledast(r,a)|) = 1$ **then**
      if $|\circledast(r,s)| = 1$ **then**
        $c_2 \Leftarrow Case(Any(\circledast(r_2, s))$;
        $t \Leftarrow Task(Any(\circledast(r_2, s))$;
        $M \Leftarrow addMit(M, NewAction(), destart, t, r_2, c_2)$;
      else
        $c_2 \Leftarrow Case(Any(\circledast(r_2, a))$;
        $t \Leftarrow Task(Any(\circledast(r_2, a))$;
      $M \Leftarrow addMit(M, NewAction(), deallocate, t, r_2, c_2)$;
      $TimeN_G \Leftarrow TimeN_G \oplus \{((n, allocated), curr())\}$;
      $TimeN_G \Leftarrow TimeN_G \oplus \{((n, started), curr())\}$;
      $ResN_G \Leftarrow ResN_G \oplus \{((n, allocated), r)\}$;
      $ResN_G \Leftarrow ResN_G \oplus \{((n, started), r)\}$;
      $M \Leftarrow addMit(M, NewAction(), destart, TaskN(n), r_1, c)$;
      $M \Leftarrow addMit(M, NewAction(), deallocate, TaskN(n), r_1, c)$;
      $M \Leftarrow addMit(M, NewAction(), allocate, TaskN(n), r, c)$;
      $M \Leftarrow addMit(M, NewAction(), start, TaskN(n), r, c)$;
  return (G, M)

back). The rollback action is our most powerful action and has a cost of 9, obtained by adding the absolute values of all the actions introduced until now. In our Payment subprocess, we can use this action when we execute a large number of updates on the same Payment Order.

Skip This action (see Algorithm 11) marks an unoffered and skippable task as 'to be skipped'. If there exists a task $t \in$ *skippable* which does not have any work item active or completed, and there not exists a mitigation action $a \in \mathcal{A}$ which skipped task t for case c, then we define $skipTask(c, G)$. To limit the use of this action, since this action may produce inconsistency in the data, we decided to assign a cost of 9. The utility of this action can be seen in two situations when we consider our running example. The first situation is the order unfulfillment. In this case, to prevent the reiterated execution of an update, we may decide to skip the "Update Shipment Payment Order" task. The second situation is the

overtime process risk. In this case we may decide to skip some tasks in order to complete the process in time.

Relocate Resource This action (see Algorithm 12) looks for a resource that is only involved in the execution of a work item belonging to a case for which no risk was defined. If once such a resource is found, it deallocates (and destarts if necessary) the work item associated with this resource and allocates the resource to a work item of the process case that we want to mitigate. The cost of this action is 7 since this action performs a (partial) sequence of destart and deallocate on two work items, and another allocate and a start action on one work item. Let x be an active border work item $x \in \circlearrowleft_G$ in case c, r be a resource involved only in case c_2, and c_2 be process which is not risky. If resource r only started or allocated one work item (of any active border events), then we can execute $relRes(c, G, M, SRC)$.

6.5 Software Implementation

We implemented the PRSA algorithm[1] as a custom service in the YAWL system.

The risk mitigation service interacts with the *Monitoring Service* that we discussed in Chapter 4, for the sake of identifying risks and computing their probabilities. It uses as input a reference to the process instance whose risks need to be mitigated, the complete YAWL specification for this instance, a log of the process (as extracted from the YAWL system), and a copy of the risk sensors associated with the process instance, as provided by the risk detection service. Modifications that a mitigation may introduce are communicated to the risk detection service, which recomputes the risk probabilities. The final solutions are returned to the user as recommendations. The one chosen by the user is then applied to the process instance under exam using the APIs provided by the YAWL engine. We implemented compensation actions associated with rolled back work items via the YAWL Worklet mechanism [HAAR10]. Accordingly, we equipped the YAWL Editor with an interface to allow users to associate a Worklet containing a compensation action to a task. When an instance of this task is rolled back, the associated Worklet is run as a separate process instance in the YAWL engine, so that from an engine perspective, the Worklet and its invoking processes are two distinct cases.

[1] Available at `https://risk-aware-bpm.googlecode.com/`

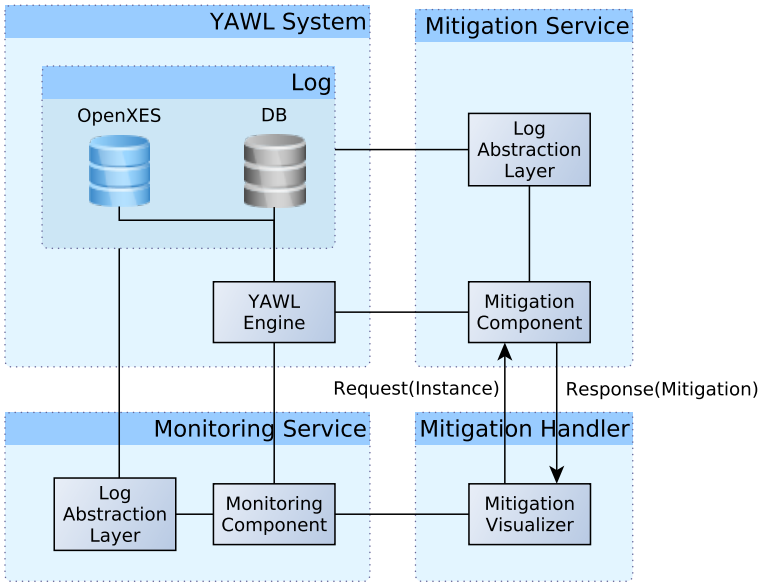


Figure 6.2: The integration of the risk mitigation tool with the YAWL system.

6.6 Evaluation

To prove the feasibility of our technique, we ran three experiments. First, we tested the required time to mitigate the same set of risks on different process models. Second, we checked the dependency of the mitigation time on different variables. Third, we checked the quality of the mitigations proposed on a specific process model.

For the first experiment, we used four real-life process models available to the research team, for which we could identify risk conditions. The sizes of these models range from 5 to 20 tasks. The first model (Process A) describes a film production process, carried out on a daily basis. This process is taken from a case study we conducted in collaboration with the Australian Film, Television and Radio School [OLH+08]. The other three models are subprocesses of an order fulfillment process. The first one (Process B) deals with the ordering, the second (Process C) deals with the payment for the goods and is the process we showed in Section 6.1, the third model (Process D) deals with the delivery of the goods.

Next, we defined seven generic risk conditions that are applicable to all these processes, where with "generic" we mean risks that are not linked to a specific context, such as a financial frauds, but that are linked to control-flow aspects. These conditions represent possible undesirable situations that may arise in a process, and relate to different process aspects such as data, resources and control-flow elements. They are domain-independent so that we could define them on all

Process	**Size**	**Variants**	**Risks avg/max**
Process A	20	127	3.53 / 7
Process B	5	7	1.71 / 3
Process C	15	31	3.05 / 5
Process D	5	15	2.13 / 4
Total	45	180	3.18 / 7

Table 6.1: Size, Variants, and Number of risks present in the processes used for the experiment.

four process models. The first condition detects a situation where two concurrent work items may not complete in a desired order. The second one is used to detect a violation of the four-eyes principle between parallel work items. The third one detects whether a time limit is exceed when executing a loop. The fourth condition detects a possible delay with the execution of a work item. The fifth one detects the possibility that two concurrent work items that should be executed by the same resource are actually allocated to two different resources (a situation that is not possible to enforce with many workflow management systems). The sixth one detects a delay with the execution of a portion of the process while the seventh one detects a data error, specifically if the data values produced by two concurrent work items are not the same.

For each process model we generated a variant with a specific combination of the above risk conditions (see Table 6.1). This led to a total of 180 process models (not every risk identified could be applied to every process model, since some of these risk conditions require parallelism and/or loops that are not present in every process model). These process models are as follows: 19 models with 1 risk condition, 40 models with 2 risk conditions, 50 with 3 risk conditions, 41 models with 4 risk conditions, 22 process models with five risk conditions, seven process models with six risk conditions, and one process model with all seven risk conditions.

For each process model we ran ten tests and averaged the results. Each test was executed on the first state of a process instance where all the risk conditions evaluated to true. For each group of tests on the same process model we measured the time required to obtain the first solution that mitigates all risks, and the number of candidate solutions generated by the algorithm in order to obtain this solution. We performed the tests on an Intel Core I5 M560 2.67GHz processor with 4GB RAM, running Linux Lubuntu v11.10.

Table 6.2 shows the results of this experiment. The second, third and fourth

Process	Mitigation time [sec]			Candidates			Variants Mitigated
	min	max	avg	min	max	avg	
Process A	0.003	178.891	26.415	2	20,181	3,456	127/127
Process B	0.001	0.033	0.015	3	54	32	7/7
Process C	0.001	0.117	0.030	2	256	60.93	31/31
Process D	0.004	0.929	0.170	2	553	78.2	15/15
Total	0.001	178.891	18.657	2	20,181	2,457	180/180

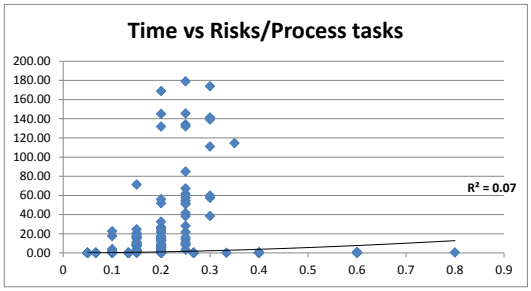
Table 6.2: Times, number of candidate solutions explored to find the first solution, and number of variants mitigated (over the total number of variants) using the PRSA algorithm.

Process	Mitigation time [sec]			Variants Mitigated
	min	max	avg	
Process A	0.059	0.235	0.113	24/127
Process B	0.043	0.065	0.054	2/7
Process C	0.023	0.052	0.035	13/31
Process D	0.028	0.101	0.061	8/15
Total	0.023	0.235	0.080	47/180

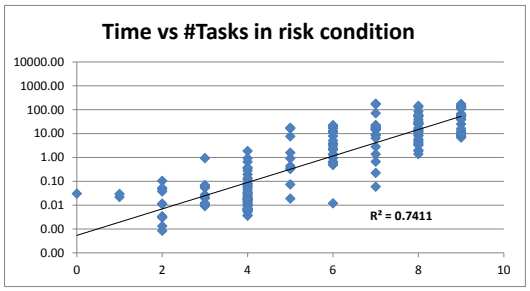
Table 6.3: Times and number of variants mitigated (over the total number of variants) using the greedy algorithm.

columns show the size (as number of tasks), the number of variants and the number of risk conditions for each of the four process models. The fifth and sixth columns show the mitigation time required to find the first solution, and the number of candidate solutions explored to find such a solution. From this table we can observe that the algorithm takes at most 3 mins (179 secs) to mitigate multiple risks in a variant of Process A (this timing refers to a combination of 5 risks for this process), though the average time is much lower (19 secs across all models). It seems reasonable to assume that in most business scenarios mitigation times in the order of a few minutes are acceptable, compared to the average time required to perform a task, and thus the average duration of a process instance. For example, let us assume an average duration of 24 hours for the Payment subprocess, with a new task being executed every 30 mins. Let us also assume that we sample the risk conditions every 5 mins. This means we have up to 6 mins to mitigate all identified risks before a new task is executed which may change the risk conditions.

Table 6.2 also shows that the algorithm needs to explore a very large number of candidate solutions to find the first solution (2,456 solutions on average across all models). While it is not fair to compare the computation power of a machine



(a) Correlation between time and risks/tasks ratio.



(b) Correlation between time and tasks in risk conditions

Figure 6.3: Correlations between time and risks/tasks ratio and between time and tasks in risk conditions.

to that of humans, this result highlights the complexity of finding a solution. It is reasonable to think that many of these candidate solutions explored by the algorithm would also need be evaluated by a human in order to find the right solution.

We also compared the results obtained using the PRSA algorithm with the results obtained using the greedy algorithm. In order to select the best mitigation, the greedy algorithm starts from the current state and selects the mitigation action that reduces the risk level the most. Once the mitigation action has been applied, it searches for another mitigation action until the combined process risk cannot be reduced further. Table 6.3 shows the results obtained using the greedy algorithm. We can notice that this algorithm, despite being on average faster than the PRSA algorithm, could not mitigate over 70% of the variants (47 over 180 variants). This result is not unexpected since a mitigation may require several mitigation actions that taken individually may not necessarily provide the best mitigation, but that may do so when combined. No differences can be detected among instances mitigated by the greedy algorithm and by the PRSA algorithm, this is due to the simplicity of the mitigations required.

In the second experiment, we investigated the factors affecting the performance of the algorithm. One would think that the mitigation time is proportional to the number of risks defined in a process model, and to the model size itself. The larger the number of risks and/or the model size, the longer it should take to mitigate such risks. However the data we extrapolated from Table 6.2 does not confirm this hypothesis. For example, the 21 variants of Process A with 5 risks have mitigation times ranging from 3.3 to 179 secs, despite their sizes and number of risks being the same. To verify that the mitigation time is not sensitive to the number of risks, nor to the process size, we plotted the correlation

Solutions [at 1 min]	**1**	**2**	**3**	**4**	**5**
Overtime Process	+	+	+	+	+
Approval Fraud	+	+	+	+	+
Order Unfulfillment	+	+	±	±	−
Cost	50	50	40	40	19

Table 6.4: Mitigations for the Payment subprocess.

between the mitigation time and the ratio risks/process size in Figure 6.3a (the solid line is the linear regression of the points). The low value of the coefficient of determination R^2 (0.07) confirms this intuition. We then checked the correlation between the mitigation time and the number of tasks used in risk conditions. The intuition is that the more work items of these tasks are pending in a given state of the process instance, the larger the number of possible mitigation actions. The corresponding scatter plot is shown in Figure 6.3b, which indeed confirms this intuition ($R^2 = 0.74$).

Finally, we checked the feasibility of the solutions proposed by the algorithm, when mitigating the domain-specific risks associated with the Payment subprocess (cf. Section 6.1). We recall that two of these risks (overtime process and order unfulfillment) are detected when the associated probability, obtained by analyzing historical data, exceeds a tolerance threshold, whereas the third risk (approval fraud) involves a complex risk condition. We considered the first state of an instance of the Payment subprocess when all three risks are active. This occurs after executing "Update shipment payment order" for the third time, once task "Approve shipment payment order" has been allocated to a resource who has already executed this task in the past.

To obtain a small number of solutions, we stopped the algorithm after one min of execution. In this time-frame, five solutions were retrieved. For each solution, Table 6.4 reports whether the solution mitigates each of the three risks, and the cost of the solution in terms of mitigation actions performed on the initial process instance. In particular, a "−" indicates a risk not mitigated, a "+" indicates a risk mitigated (with risk probability lower than the specific threshold if the condition depends on the risk probability), and a "±" indicates a risk mitigated whose condition cannot be computed for lack of information, i.e. some of the variables used in the risk condition are null. We recall that the algorithm prioritizes a solution whose risk is mitigated by computing the risk condition, than a solution whose risk is mitigated because the respective condition cannot be computed.

The five solutions identified are pairwise mutually non-dominating. Solutions

1 and 2 are dominated by solutions 3, 4 and 5 cost-wise, but dominate these solutions w.r.t. the mitigation of the order unfulfillment risk. Solution 5 dominates solutions 3 and 4 cost-wise but is dominated by these two solutions w.r.t. the mitigation of the order unfulfillment risk.

Let us briefly examine the mitigations performed by the five solutions. The first four solutions mitigate the approval fraud by deallocating the resource that was allocated "Approve shipment payment order", while solution 5 additionally allocates the work item to a resource who did not execute this task for the same customer in the past. All these mitigations are feasible, though the one provided by solution 5 is more robust, since there is no risk that the task gets allocated to a resource who has already executed it. The order unfulfillment risk is mitigated by solutions 1 and 2 through rolling back the work item of task "Update shipment payment order" (which leads to a deoffer of the work item of task "Approve shipment payment order" that comes afterwards). Solutions 3 and 4 do this too but also mark this task 'to be skipped' preventing a possible re-execution of it. This action sets to null the risk variables associated with this task that retrieve the number of executions and its estimated remaining time making the risk mitigated but not computable. Thus, while all four solutions are feasible, we would prioritise the first two since these ensure that the risk probability has actually dropped below the threshold. Finally, all solutions differ in the way they mitigate the overtime process risk. Each of them skips a different task among those not yet executed (for simplicity, all of them have the same estimated duration). Despite the fact that all these solutions are feasible, only the mitigation proposed by solution 3 is interesting since it proposes to skip tasks "Update Shipment Payment Order" and "Approve Shipment Payment Order" avoiding this way that the loop is taken again. In other words, it prevents the order to undergo further updates, and subsequent approvals.

6.7 Related Work

The mitigation actions proposed in this chapter share commonalities with the workflow exception patterns [RAH06]. For example, the *reoffer* pattern at the work item level can be obtained by combining our mitigation actions *deoffer* + *offer*, which represent atomic operations in this respect. Another similarity is between our *rollback* action and the recovery patterns *rollback* and *compensate*, since our rollback can also trigger a compensation action if this is available for the task being rolled back. Given that we need to operationalize these actions,

we do provide a precise characterization of all actions and their effects, whereas the work of Russell et al. [RAH06] limits itself to a textual description of these exception patterns, as they are observed from an analysis of process modeling languages and tools. That said, the main contribution of our work is not the proposed set of mitigation actions per se (which could be replaced or modified) but rather an automated mechanism for combining these actions in an optimal way in order to mitigate a set of process-related risks as far as possible.

Risks like those we illustrated in this chapter can also be encoded as *constraints* on top of a process model. Approaches exist that can check whether there exists at least an instance of a constrained process model that can be executed without violating the constraints. For example, Combi and Posenato [CP09] propose a general method for checking the satisfiability of temporal process constraints (like our overtime process risk) at design-time. Other approaches can also enforce these constraints at run-time, so that any process instance will satisfy all the constraints by construction. For example, Tan et al. [TCG04] propose a model for constrained workflow execution that addresses cardinality constraints (e.g. to control how many times a task can be executed), binding of duty constraints (i.e. the ability of a resource to retain a familiar work item) and separation of duty constraints (i.e. different resources should execute different tasks). These constraints can be enforced only within a given process instance. The approach proposed by Warner and Atluri [WA06] overcomes this limitation by proposing a constraint specification language for resource allocation that also addresses inter-instance constraints. Compared to our work, constrained workflow execution provides a rigid approach according to which if the constraints cannot be satisfied, a process model will simply not be instantiated, or if instantiated, the instance violating the constraints will throw an exception. Commercial systems such as IBM WebSphere and AristaFlow also support constrained workflow execution, and handle these violations (e.g. violations of temporal constraints) by escalating control to a process administrator.[2] Instead, our technique allows a process instance to be automatically *adapted* on-the-fly in order to reduce the risk of violating such constraints.

Various research frameworks have been proposed for the *dynamic adaptation* of process instances. For example, ADEPT [DR09] supports adding, deleting and changing the sequence of tasks at both the model and instance levels, however such changes must be achieved via manual intervention by an administrator.

[2] `http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r1mx/index.jsp?topic=/com.ibm.wbit.612.help.tel.ui.doc/topics/tescal8.html`

AgentWork [MGR04] provides the ability to modify process instances by dropping and adding individual tasks based on events and rules. CBRFlow [WWB04] uses case-based reasoning to support runtime adaptation by allowing users to annotate rules during process execution. CEVICHE [HSD10] is a service-based framework that uses the AO4BPEL (Aspect-Oriented for BPEL) language [CM07] to provide an option for skipping or reallocating tasks to other services in an ad-hoc manner. While these approaches could be used for risk mitigation purposes, they do not provide any help for the identification of which particular mitigation actions should be used. The YAWL Worklet Service [AHAE07] provides each task of a process instance with the ability to be associated with an extensible repertoire of actions ('drop-in' processes), one of which is contextually and dynamically bound to the task at run-time. It also supports capabilities for dynamically detecting and handling run-time exceptions. However the approach is generic and not specifically designed for risk detection and mitigation. Also a new situation cannot automatically be dealt with but requires a workflow administrator to intervene.

Our work is also related to *operational support* in process mining [Aal11]. Operational support deals with the analysis of current and historical execution data, with the aim to predict future states of a running process instance, and provide recommendations to guide the user in selecting the next activity to execute based on certain objectives. For example, the approach for cycle time prediction presented by Van der Aalst et al. [ASS11] could be, with the opportune modifications, adapted for risk prediction. Using this approach it would be possible to estimate the probability of an overtime risk and suggest the next steps the current instance should take in order to keep this risk under control. The application of this approach unfortunately requires that the process model captures all the possible mitigation actions as normal activities, i.e. as control-flow alternatives. For instance, if a task can be skipped, there should be a path without that task that leads to the end node of the process model. This may drastically increase the complexity of the process model. Moreover, this approach would not be applicable to capture mitigation operations on resources (i.e. deallocating a resource) or on task states (e.g. suspending a task). That said, more in general, our technique could be seen as a possible provider for operation support, and could thus be integrated in process mining environments like ProM.[3]

Our work provides recommendations to users as to which mitigation actions can be applied to the specific context at hand. As such, it shares commonalities

[3] `http://processmining.org`

with recommendation and decision support systems. Alter [Alt04] states that the focus of such systems should be towards improving decision making within work systems, rather than externalizing support. This view is shared by our technique, which provides an extension to existing process-aware information systems, rather than a separate standalone tool. As such, it may be considered a member of the domain known as *Group Decision Support Systems*, which facilitate task support in group environments.

Gambini et al. [GLMH11] explored the use of dominance-based MOSA for automatically fixing behavioral errors in process models, at design-time. Our work on risk mitigation can thus be seen as an adaptation of that idea to run-time aspects, since we aim to improve running process instances. Besides their distinct aims, the main difference between the two approaches is that for correcting behavioral errors they defined three objective functions capturing the structural and behavioral similarity of a solution to the incorrect model, whereas in risk mitigation the number and type of objective functions depends on which risks are active in a given state of a process instance.

6.8 Summary

This chapter contributes a concrete technique for the automatic mitigation of process-related risks at run-time. The technique requires as input an executable process model and a set of associated risk conditions. At run-time, when one or more risk conditions evaluate to true, a process administrator can launch our technique to mitigate the identified risks and bring the process instance back to a safe state. This is achieved by generating a set of possible mitigations that change the current instance in order to bring the likelihood of the identified risks below a tolerance level. These mitigation actions are not performed directly on the instance under consideration. Rather, their effects are simulated and those solutions that mitigate the most risks in a given time-frame, are proposed as recommendations to the process administrator.

The mitigation actions are determined via a dominance-based multi-objective simulated annealing algorithm. This choice allows us to explore the solution space as widely as possible, avoiding local optima. In essence, each risk is treated as an objective function whose likelihood needs be minimized. The objective is reached as soon as the likelihood goes below the tolerance value for that particular risk. Mitigation actions affect various aspects of a process, such as task execution and resources utilization. To the best of our knowledge, this is the first time that

process-related risks can be mitigated automatically.

The technique was implemented in the YAWL system and its performance evaluated with real-life process models. The tests show that on the analyzed process models a set of possible solutions can be found in a matter of seconds, or within a few minutes in the worst case, and that in all cases the associated risks are mitigated. We expect this technique to reduce the effort and time required by process administrators to understand what mitigation actions are feasible based on a particular state of the system.

Chapter 7

Conclusion

This thesis presented a risk-aware business process management framework to address three shortcomings in the context of risk-aware business process management: (i) lack of adequate support for the detection of the eventuation of process-related risks during process execution; (ii) lack of support for the prediction of such risks and for their prevention; and (iii) lack of support for their mitigation.

The first shortcoming was addressed through the realization of a technique for real-time monitoring of risks in executable business process models. The technique embeds elements of risk into the BPM lifecycle: from process design, where high-level risks are mapped to specific process model elements, through to process diagnosis, where risks are detected during process execution and process administrators are notified of those risks that are no longer tolerable. To the best of our knowledge, this is the first solution that concretely embeds risks into executable business processes and enables their automatic detection at run-time.

The proposed technique was operationalized through the use of a distributed sensor-based architecture. Each risk is associated with a sensor condition, which is an undesired state of the process. Conditions can relate to any process aspect both from the current process instance and from completed instances. At design-time, these conditions are expressed within a process model via a simple query language, or through the use of predefined templates. At run-time, when a condition is violated, a notification is sent to a process administrator about the given risk.

In order to prevent process risks, the second noted shortcoming, we developed a recommendation system that allows process participants to take risk-informed decisions. Using historical information extracted from process execution logs, for each state of a process instance where input is required from a process participant, the system determines the risk that a fault (or set of faults) will occur, if the

participant's input is going to be used to carry on the process instance. This input can be in the form of data used to fill out a user form, or in terms of the choice of which work item to execute next.

The system relies on two techniques: one for predicting risks, the other for identifying the best assignment of participants to the work items currently on offer. The objective is to minimize both the overall risk (i.e. the combined risk for all faults) and the execution time of all running process instances.

We designed the system in a language-independent manner, using common notions of executable process models such as tasks and work items. We then implemented the system as a set of components for the YAWL system. Although we implemented our ideas in the context of the YAWL system, our recommendation system can easily be integrated with other BPM systems.

The third and last shortcoming was addressed through a technique for the automatic mitigation of process-related risks at run-time. The technique requires as input an executable process model and a set of associated risk conditions. At run-time, when one or more risk conditions evaluate to true, a process administrator can use our technique to mitigate the identified risks and bring the process instance back to a safe state. This is achieved by generating a set of possible mitigations that change the current instance in order to bring the likelihood of the identified risks below a given tolerance level. These mitigation actions are not performed directly on the instance under consideration. Rather, their effects are simulated and those solutions that mitigate the majority of risks in a given time-frame, are proposed as recommendations to the process administrator.

The mitigation actions are determined via a dominance-based multi-objective simulated annealing algorithm. This choice allows us to explore the solution space as widely as possible, avoiding local optima. In essence, each risk likelihood is treated as an objective function that needs to be minimized. The objective is reached as soon as the likelihood goes below the tolerance value for that particular risk. Mitigation actions affect various aspects of a process, such as task execution and resource utilization.

In relation to our solution criteria discusses in the Introduction, the approach presented in this thesis addresses all identified criteria for the management of process-related risks in executable business processes, i.e. in those processes that are directly executed by a BPMS or supported by an information system that produces event logs. The approach is supported by formal foundations and each of its components has been formally defined. Moreover, while it is realized on top of a specific system, i.e. the YAWL BPMS, it does not depend on this system and can

easily be adapted to different process modelling languages and systems. Finally, each of the techniques here presented has been prototyped and evaluated with real-life and artificial data. The response times of these techniques are between a few milliseconds and a few minutes, which make them suitable for near real-time applications [Wer78]. Further, the usability and easy of use evaluation of the risk monitoring component, showed that the technique provided value to develop an understanding of process risks, particularly for users with limited experience in process modeling or BPMSs.

The work discussed in this thesis suffers from several limitations which offer opportunities for future work. Firstly, to evaluate the perceived ease of use of the sensor-based architecture we only checked if the users were able to interpret a predefined risk condition; we did not ask them to model risks and this can be an interesting avenue for future work. Secondly, to overcome the low perceived ease of use of the sensor-based architecture resulting from the usability tests, the concrete syntax of the sensor definition language can be improved taking into account the design principle of clarity. Thirdly, manual work is currently required to convert the results of a risk analysis techniques (e.g. fault trees) into sensor conditions. This operation is error-prone, especially in the context of complex risk definitions. In this respect, a mechanism for automatically deriving skeletons of sensor conditions directly from a fault tree can be devised. This will allow a smoother transition from high-level risk definitions to low-level sensor conditions. Moreover, aiding features provided by common source code editors such as autocomplete, syntax highlighting and bracket matching can be put in place to reduce the human effort. Finally, the usefulness of templates with end users needs to be evaluated. In doing so, it is important to consider the results against their correlation with the user background in terms of exposure to YAWL in particular, and to risk management in general.

The recommendation system we presented relies on a couple of assumptions. While we deal with multiple process instances sharing the same pool of participants, we assume no sharing of data between instances. Further, we assume that a participant can perform a single task at a time. These assumptions can be lifted. For example, for the sharing of data between instances the ILP problem needs to be reformulated in order to consider that the risk estimation of a work item may change as a consequence of the modification of data by work items that have been scheduled to be performed first. In order to allow participants to perform multiple tasks at a time, it is required to assign a capacity to each resource as the maximum number of work items that a resource can perform in parallel. Our ILP

problem needs to be reformulated in order to take this capacity into account. In future work, it will also be interesting to extend this technique such that it will only provide suggestions in line with the business goals of a company. Achieving this will require the definition of a language for the specification of business goals that will then be integrated with the current recommendation system.

An other avenue for future work concerns the validation of the feasibility and appropriateness of the proposed mitigation actions with domain experts. This can be achieved by comparing the solutions obtained with our algorithm with those proposed by such experts. The exploration of the solution space can be improved by prioritizing the mitigation of those risks that have the highest impact on the process objectives. In fact, currently all risks are treated alike whereas in reality this might not be the case. Further, the algorithm could also be extended to prioritize certain mitigation actions based on how these have been ranked by users in previously mitigated instances.

Another avenue for future work is to investigate how identified risk mitigations and prediction suggestions can be incorporated into the process design, in order to create improved process specifications that are less prone to risk.

Appendix A

Detailed Abstract Syntax

This appendix provides a detailed description of each element of the abstract syntax of the sensor definition language.

Definition		**Description**
Sensor	$\triangleq$ *v* : *Variables*; *t* : *Trigger*; *risk* : *RiskCon*; *fault* : *BoolCon*; *consequence* : *MaCon*;	A sensor is composed of a set of variables, a trigger, a risk condition, a fault condition, and a consequence.
Trigger	$\triangleq$ *timer* | *event*	A trigger can either can be timer or event based
Variables	$\triangleq$ *Assignment*$^+$	A set of variables is a several assignments.
Assignment	$\triangleq$ *result* : *varName*; *i* : *Info*	A variable assigns a name to an information.
Info	$\triangleq$ *VarFun* | *Definition* | *CaseExp* | *CaseEleExp*	An information is a function invoked on a variable, a definition, or an information collected from the process instance (i.e. *CaseExp* or *CaseEleExp*).
VarFun	$\triangleq$ *ResCon* | *ResSimFun* | *ResComFun*	The three types of functions invoked on variables are available: *ResCon*, *ResSimFun* or *ResComFun*.
Definition	$\triangleq$ *c* : *constant*	A definition is a constant.
CaseExp	$\triangleq$ *cis* : *CaseIDStat*; *a* : *Action*	An information related to the process instance.
CaseEleExp	$\triangleq$ *ce* : *CaseExp*; *ton* : *TaskOrNet*	An of information related to an element (i.e. a task or a net) of the process instance.
TaskOrNet	$\triangleq$ *taskLabel* | *netName*	The identifier of a task or of a net.
CaseIDStat	$\triangleq$ *absExp* | *relExp* | *CaseConSet*	Identifies an instance or a set of instances.
Action	$\triangleq$ *predAct* | *taskOrNetVar* | *SubVarExp* | *inputPredAct*	An action is either a predefined action, a task variable, a net variable, a subvariable, or a predefined action that requires an input.
SubVarExp	$\triangleq$ *var*$^+$	A subvariable of a task variable or of a net variable.
CaseConSet	$\triangleq$ *CaseCon* | *CaseParam* | *CaseConExp*	The three types of identifiers for a set of instances.
CaseCon	$\triangleq$ *ton* : *TaskOrNet*; *a* : *Action*; *co* : *CompOp*; *rhe* : *RHExp*	Identifies instances for which a specific piece of information is compared with a *RHExp*.
CaseParam	$\triangleq$ *i* : *idFun*; *co* : *CompOp*; *rhe* : *RHExp*	Identifies instances for which the instance ID is compared with a *RHExp*.
CaseConExp	$\triangleq$ ccs_1, ccs_2 : *CaseConSet*; *bo* : *BoolOp*	Identifies instances for which several conditions are satisfied.
CompOp	$\triangleq$ *l* | *leq* | *g* | *geq* | *eq* | *contains* | *isContained*	Set of comparison operators.

Definition		Description
$RHExp$	$\triangleq$ $constant \mid function \mid varName \mid RHExpSet$	Can be a constant, a function, a function invoked on a variable, or a $RHExpSet$.
$RHExpSet$	$\triangleq$ $rhe_1, rhe_2 : RHExp; o : Op$	An arithmetical expression built using $RHExp$.
$RiskCon$	$\triangleq$ $likelihood, threshold : MaCon$	A risk condition is composed of two $MaCon$.
$MaCon$	$\triangleq$ $MaITE \mid MaExp \mid MaFor \mid FixMaFor \mid ResSimFun \mid ResComFun \mid varName \mid constant$	A arithmetical condition can be an if-then-else expression, an expression, a loop, a function on a variable, a variable name, or a constant.
$MaITE$	$\triangleq$ $if : BoolCon; then, else : MaCon$	A arithmetical condition that varies based on the result of a boolean condition.
$MaExp$	$\triangleq$ $MaUnExp \mid MaBinExp$	The two types of arithmetical expression: $MaUnExp$ or $MaBinExp$.
$MaUnExp$	$\triangleq$ $s : sub; me : MaCon$	A negative arithmetical expression.
$MaBinExp$	$\triangleq$ $me_1, me_2 : MaCon; mo : MaOp$	A arithmetical operation between $MaCon$s.
$MaOp$	$\triangleq$ $add \mid sub \mid mul \mid div \mid exp \mid mod$	The set of arithmetical operators.
$BoolCon$	$\triangleq$ $BoolITE \mid BoolExp \mid BoolFor \mid FixBoolFor \mid Comp \mid ResCon \mid varName \mid constant$	A boolean condition can be an if-then-else expression, an expression, a loop, a comparison, a function on a variable, a variable, or a constant.
$BoolITE$	$\triangleq$ $if, then, else : BoolExp$	A boolean condition that varies based on the result of another boolean condition.
$BoolExp$	$\triangleq$ $BoolUnExp \mid BoolBinExp$	The two types of boolean expression: $BoolUnExp$ or $BoolBinExp$.
$BoolUnExp$	$\triangleq$ $n : neg; e : BoolCon$	A negated boolean expression.
$BoolBinExp$	$\triangleq$ $e_1, e_2 : BoolCon; bo : BoolOp$	A boolean operation between $BoolCon$s.
$BoolOp$	$\triangleq$ $and \mid or$	The set of boolean operators.
$Comp$	$\triangleq$ $ce_1, ce_2 : CompElem; co : CompOp$	A comparison between two $CompElem$s.
$CompElem$	$\triangleq$ $MaCon \mid ResListFun \mid varName \mid constant$	A comparison element can be a arithmetical condition, a function invoked on a variable, a variable, or a constant.
$ResCon$	$\triangleq$ $res : varName; a : Activity$	A function invoked on a variable using an activity.
$ResListFun$	$\triangleq$ $result : varName; lrf : listResFun$	A function invoked on a variable. It returns a list.
$ResSimFun$	$\triangleq$ $resource : varName; srf : simResFun$	A function invoked on a variable which is a resource.
$ResComFun$	$\triangleq$ $res_1, res_2 : varName; crf : comResFun$	A function invoked on two variables which are resources.
$Activity$	$\triangleq$ $resource : varName; lrf : ResListFun$	A function invoked on a variable which is a resources.
$MaFor$	$\triangleq$ $t : TypeMF; lr : ListResult; fme : ForMaCon$	A nested loop that cycles on each element of a $ListResult$ and aggregates the result of a $ForMaCon$ based on a $TypeMF$.
$FixMaFor$	$\triangleq$ $fixEle : varName; mf : MaFor$	A $MaFor$ that loops on each element of a $ListResult$ using as index the element $fixEle$.
$TypeMF$	$\triangleq$ $add \mid mul$	Two forms of aggregations: adding and multiplying.
$BoolFor$	$\triangleq$ $t : TypeBF; lr : ListResult; fbe : ForBoolCon$	A nested loop that loops on each element of a $ListResult$ and aggregates the result of a $ForBoolCon$ based on a $TypeBF$.
$FixBoolFor$	$\triangleq$ $fixEle : varName; bf : BoolFor$	A $BoolFor$ that loops on each element of a $ListResult$ using as index the element $fixEle$.
$TypeBF$	$\triangleq$ $and \mid or$	Two forms of aggregations: AND or OR.
$ListResult$	$\triangleq$ $varName^+$	A list of variables.

Definition		Description
$ForMaCon \triangleq$	$ForMaITE \mid ForMaExp \mid constant \mid varName$	A arithmetical condition can be an if-then-else expression, an expression, a constant, or a variable.
$ForMaITE \triangleq$	$if : ForBoolCon; then, else : ForMaCon$	A arithmetical condition that varies based on the result of a boolean condition.
$ForMaExp \triangleq$	$ForMaUnExp \mid ForMaBinExp$	The two types of arithmetical expression: *ForMaUnExp* or *ForMaBinExp*.
$ForMaUnExp \triangleq$	$s : sub; me : ForMaCon$	A negative arithmetical expression.
$ForMaBinExp \triangleq$	$me_1, me_2 : ForMaCon; mo : MaOp$	A arithmetical operation between *ForMaCon*.
$ForBoolCon \triangleq$	$ForBoolITE \mid ForBoolExp \mid ForComp \mid varName \mid constant$	A boolean condition can be an if-then-else expression, an expression, a comparison, a variable, or a constant.
$ForBoolITE \triangleq$	$if, then, else : ForBoolExp$	A boolean condition that varies based on the result of another boolean condition.
$ForBoolExp \triangleq$	$ForBoolUnExp \mid ForBoolBinExp$	The two types of boolean expression: *ForBoolUnExp* or *ForBoolBinExp*.
$ForBoolUnExp \triangleq$	$n : neg; e : ForBoolCon$	A negated boolean expression.
$ForBoolBinExp \triangleq$	$e_1, e_2 : ForBoolCon; bo : BoolOp$	A boolean operation between *ForBoolCon*s.
$ForComp \triangleq$	$ce_1, ce_2 : ForCompElem; co : CompOp$	A comparison between two *ForCompElem*s.
$ForCompElem \triangleq$	$ForMaCon \mid varName \mid constant$	A comparison element can be a arithmetical condition, a variable, or a constant.

Appendix B

Actions

This appendix describes all actions available in the sensor definition language.

Action	**Description**
(ID)	returns the ID of the generic instance that is being analyzed
[IDCurr]	returns the ID of the instance that the sensor is monitoring
Count	returns the number of times a task has been completed
offerResource	returns the resources to which the task has been offered
allocateResource	returns the resources to which the task has been allocated
startResource	returns the resources that started the task
completeResource	returns the resource that completed the task
isOfferd	returns "*true*" if the task has been offered
isAllocated	returns "*true*" if the task has been allocated
isStarted	returns "*true*" if the task has been started
isCompleted	returns "*true*" if the task has been completed
OfferTime	returns the time when the task has been offered
AllocateTime	returns the time when the task has been allocated
StartTime	returns the time when the task has been started
CompleteTime	returns the time when the task has been completed
OfferTimeInMillis	returns the time (in millisecond) when the task has been offered
AllocateTimeInMillis	returns the time (in millisecond) when the task has been allocated
StartTimeInMillis	returns the time (in millisecond) when the task has been started
CompleteTimeInMillis	returns the time (in millisecond) when the task has been completed
PassTimeInMillis	returns the amount of time (in millisecond) that was needed to complete the task
TimeEstimationInMillis	returns an estimation of the time (in millisecond) needed to completed the task/process
Variable	returns the value of the variable or sub-variable required
offerDistribution	returns the list of resources to which the task is offered by default
allocateDistribution	returns the list of resources to which the task is allocated by default
startDistribution	returns the list of resources to which the task is started by default
offerInitiator	returns the offering policy of the task (user or system)
allocateInitiator	returns the allocating policy of the task (user or system)
startInitiator	returns the starting policy of the task (user or system)
CountElements	returns the number of instances that satisfy the parameters required
FraudProbabilityFunc	returns the probability of a fraud using as parameters: the current number of executions, the maximum number of executions allowed, the parameter used to group these instances, the parameter used to identify a temporal window, and the dimension of the temporal window

List of actions of the architecture

Appendix C

Nested Loops

This appendix describes the four types of nested loops available in the sensor definition language.

For	Description
forAND[][]	executes a nested loop for each list provided in input (among the first couple of brackets) resolving the expression (defined among the second couple of brackets), then returns the AND conjunction of the results obtained
forOR[][]	executes a nested loop for each list provided in input (among the first couple of brackets) resolving the expression (defined among the second couple of brackets), then returns the OR conjunction of the results obtained
forADD[][]	executes a nested loop for each list provided in input (among the first couple of brackets) resolving the expression (defined among the second couple of brackets), then returns the sum of the results obtained
forMUL[][]	executes a nested loop for each list provided in input (among the first couple of brackets) resolving the expression (defined among the second couple of brackets), then returns the product of the results obtained

List of nested loops

Appendix D

Functions

This appendix describes the functions available in the sensor definition language.

Function	**Description**
offeredList	returns the list of tasks currently offered to the resource/resources (returns a list of list if used on resources)
allocatedList	returns the list of task currently allocated to the resource/resources (returns a list of list if used on resources)
startedList	returns the list of task currently started by the resource/resources (returns a list of list if used on resources)
offeredNumber	returns the number of tasks currently offered to the resource/resources (returns a list if used on resources)
allocatedNumber	returns the number of tasks currently allocated to the resource/resources (returns a list if used on resources)
startedNumber	returns the number of tasks currently started by the resource/resources (returns a list if used on resources)
offeredMinNumber	returns the minimum number of tasks currently offered to the resource/resources
allocatedMinNumber	returns the minimum number of tasks currently allocated to the resource/resources
startedMinNumber	returns the minimum number of tasks currently started by the resource/resources
offeredMinNumberExcept	returns the minimum number of tasks currently offered to the resource/resources excluding the resource/resources provided in input (the input is provided using a dotted format after the name of the function)
allocatedMinNumberExcept	returns the minimum number of tasks currently allocated to the resource/resources excluding the resource/resources provided in input (the input is provided using a dotted format after the name of the function)
startedMinNumberExcept	returns the minimum number of tasks currently started by the resource/resources excluding the resource/resources provided in input (the input is provided using a dotted format after the name of the function)
offeredContain	returns *true* if the task provided in input (using a dotted format after the name of the function) is currently offered to the resource/resources.
allocatedContain	returns *true* if the task provided in input (using a dotted format after the name of the function) is currently allocated to the resource/resources.
startedContain	returns *true* if the resource/resources started the task provided in input (using a dotted format after the name of the function).

List of functions

Appendix E

Questionnaire

Part 1) Background Questions

E1a: Which description matches best your current status?

- Student
- Academic
- Professional

E1b:Please specify your gender:

- Female
- Male
- Prefer not to tell.

E2: How many years ago did you start process modeling?

_____ Years

E3a: How many process models have you analyzed or read within the last 12 months? (A year has about 250 work days. In case you read one model per day, this would sum up to 250 models per year)

_____ Models

E3b: How many process models have you created or edited within the last 12 months?

_____ Models

E3c: How many activities did all these models have on average?

_____ Activities

E4a: How many work days of formal training on process modeling have you received within the last 12 months? (This includes e.g. university lectures, certification courses, training courses. 10 weeks of a 120 minute university lecture is roughly 3 work days)

_____Days

E4b: How many work days of self-education have you made within the last 12 months? (This includes e.g. learning-by-doing, self-study of textbooks or specifications)

_____Days

Part 2) Your experience with Workflow Management Systems

Q1: Which of the following (process) modelling techniques other than YAWL have you used to describe a process or procedure? Tick all that apply.

- None. Please go to question 10.
- BPMN
- UML
- Activity Diagrams
- EPCs
- BPEL
- Petri Nets
- Protos
- Other:

Q2: If you have used any technique other than YAWL, roughly, how many **conceptual process** models do you think you have created?

_____ process models

Q3: If you have used any technique other than YAWL, roughly, how many **workflow** models (i.e. executable process models) do you think you have read?

_____ workflow models

Q4: How long have you been using a Workflow Management System?

- I am evaluating to do so/I have just started
- Less than 1 month
- 1 - 6 months
- 7 - 12 months
- More than 1 year

Q6: Indicate your level of agreement to the following statements on the given scale by circling the number that best describes your view on the statement.

	Strongly disagree	Disagree	Somewhat disagree	Normal	Somewhat agree	Agree	Strongly agree
Overall, I am very familiar with Work flow Management Systems.	1	2	3	4	5	6	7
I feel very confident in my understanding of Workflow Management Systems.	1	2	3	4	5	6	7
I feel very competent in using Workflow Management Systems.	1	2	3	4	5	6	7

Part3) Your experience with the YAWL system

Q1: How long have you been using YAWL?

- I am evaluating to do so/I have just started
- Less than 1 month
- 1 - 6 months
- 7 - 12 months
- More than 1 year

Q2: Please indicate, roughly, the typical extent of usage of YAWL. This includes any activity related to the YAWL system, e.g. development, reading documentation, modelling, executing or simulating YAWL workflows, and using a system that is based on some YAWL component. Keep in mind, a regular working days has eight hours (480 minutes).

- Not applicable
- On average, I spend _____ hours and _____ minutes on YAWL every working day.

Q3: Roughly, how many YAWL models do you think you have created or read?

_____ YAWL models

Q4: Which features of the YAWL system have you ever used? Tick all that apply

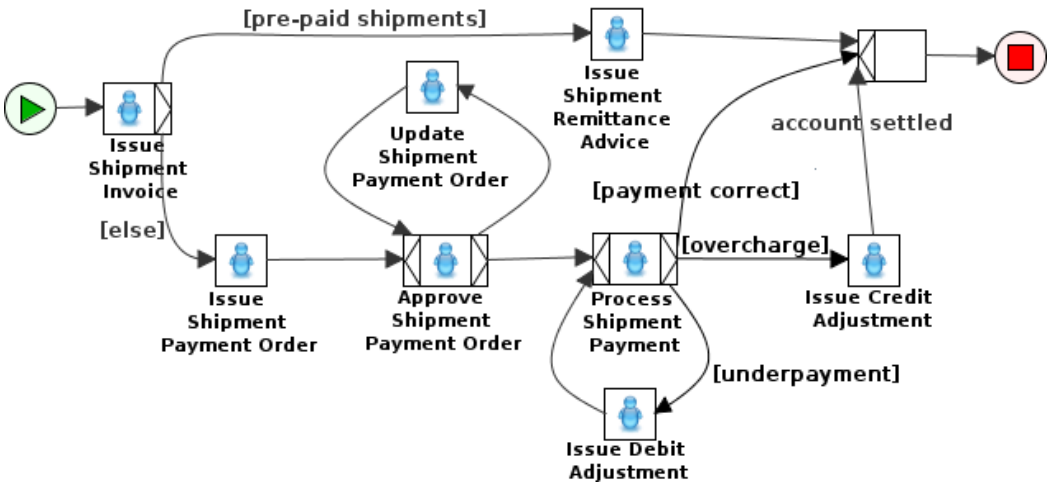
- Not applicable
- Execution environment
- Syntax checker/verification
- Cancelation region
- OR-join
- Multiple instance task
- Deferred choice
- Milestone

- Retain familiar/Separation of duties
- Deferred distribution
- Distribution set filter
- Allocation strategies
- Worklets/Exlets
- Custom services
- Configuration

Part 4) Risk Sensor Comprehension

In this part, you will be shown 3 risks related to a process described in YAWL. You will be asked questions about each of them.

Risk 1: Consider the following YAWL model and associated risk condition described below.



Variables:

A = case(current).Update Shipment Payment Order(Count)

B = case(Update Shipment Payment Order(Count)>=E).Update Shipment Payment Order(CountElements)

C = case(Update Shipment Payment Order(Count)>=A AND Process Shipment Payment(isOffered)="true").Update Shipment Payment Order(CountElements)

D = 0.6

E = 5

Sensor Condition:***(C/B)>D*** where"/" is the division operator

Q0. How difficult is it to understand the meaning of the above sensor condition:

- 1 – very simple
- 2 – rather simple
- 3 – neutral
- 4 – rather difficult
- 5 – very difficult

Q1. Does the sensor send a notification if the task Update Shipment Payment Order may not be executed?

- Yes / No / I do not know

Q2. Can the sensor notify a risk if the task Update Shipment Payment Order has been executed only twice?

- Yes / No / I do not know

Q3. Does A retrieve the value of a variable named Count of the task Update Shipment Payment Order?

- Yes / No / I do not know

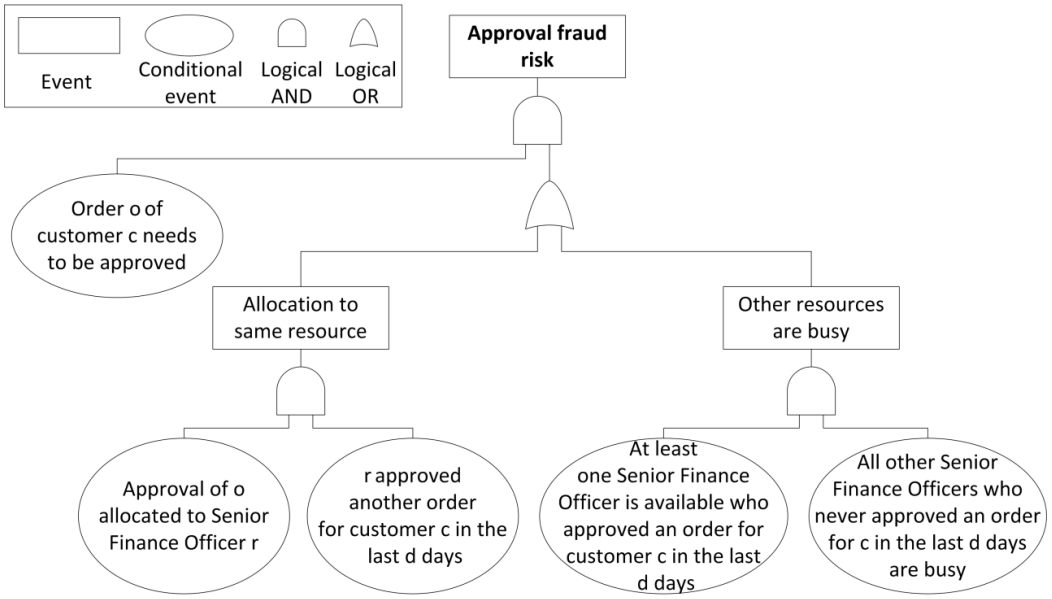
Q4.Does C return the number of instances where the task Process Shipment Payment has been offered?

- Yes / No / I do not know

Q5. Does D indicate a threshold that if exceeded produces a notification from the sensor?

- Yes / No / I do not know

Risk 2: Using the YAWL model of page 6 above, an Approval fraud risk has been identified using Fault Tree Analysis, as shown below.



To detect the risk of this fault, we first have to check whether there is an order, say order o of customer c, to be approved. This means checking that an instance of task Approve Shipment Payment Order is being executed. Moreover, we need to check that either of the following conditions holds:

1. o has been allocated to a Senior Finance Officer who has already approved another order for the same customer in the last d days; or
2. at least one Senior Finance Officer is available who approved an order for customer c in the last d days and all other Senior Finance Officers who never approved an order for c during the last d days are available

Q0. How difficult is it to understand the meaning of the above risk:

- 1 – very simple

- 2 – rather simple
- 3 – neutral
- 4 – rather difficult
- 5 – very difficult

Q1. Could condition Abe used as trigger for the sensor?

- Yes / No / I do not know

Q2. If condition B is expressed using a variable, would the following assignment be correct?

B = case(current).Approve Shipment Payment Order(allocateResource)

- Yes / No / I do not know

Q3. Could condition D be expressed using only one variable in the risk condition?

- Yes / No / I do not know

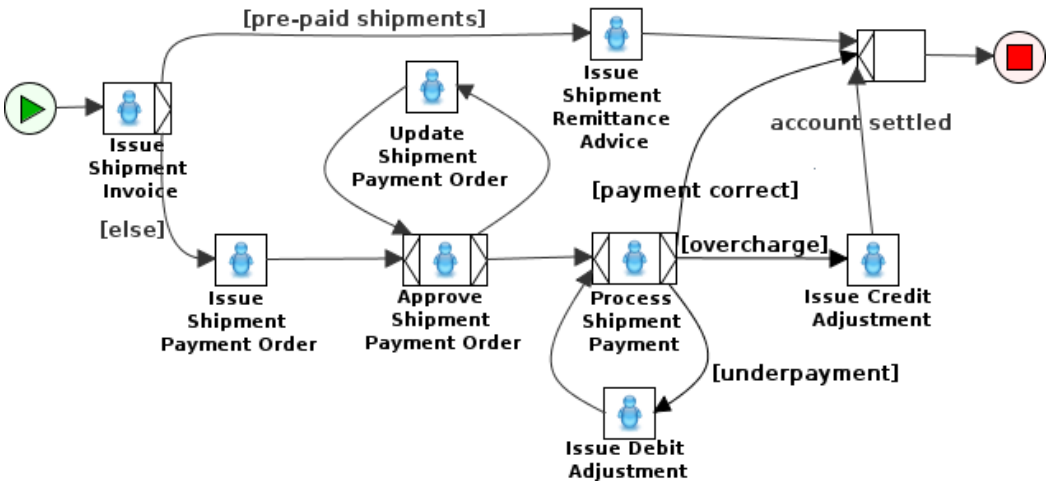
Q4. Could condition E be expressed using only one variable in the risk condition?

- Yes / No / I do not know

Q5. Could the AND between conditions B and C be expressed using only one variable in the risk condition?

- Yes / No / I do not know

Risk 3: Consider the following YAWL model and the associated risk condition described below.



Variables:

A = case(current).Process Shipment Payment.Customer

B = case(Process Shipment Payment.Customer=A AND Issue Debit Adjustment(isCompleted)="true").Issue Debit Adjustment(CountElements)

C = case(Process Shipment Payment.Customer=A AND Issue Credit Adjustment(isCompleted)="true").Issue Debit Adjustment(CountElements)

D = case(Process Shipment Payment.Customer=A).Process ShipmentPayment(CountElements)

F = 0.4

Sensor Condition:
(B/D)>F|(C/D)>F where"/" is the division operator and "|" is the logical OR operator

Q0. How difficult is it to understand the meaning of the above sensor condition:

- 1 – very simple
- 2 – rather simple
- 3 – neutral
- 4 – rather difficult
- 5 – very difficult

Q1. Does the sensor send a notification if the task Issue Debit Adjustment is not executed?

- Yes / No / I do not know

Q2. Will the sensor send a notification as soon as the task Issue Credit Adjustment is completed?

- Yes / No / I do not know

Q3. Does B return the number of instances where the customer is the same of the current instance and the task Issue Debit Adjustment has been completed?

- Yes / No / I do not know

Q4. Does C return the number of instances where the task Process Shipment Payment has been offered?

- Yes / No / I do not know

Q5. Does B always return a value greater than the value return by C?

- Yes / No / I do not know

Part 5) Your views on the use of the Sensor-based component of the YAWL system

This part of the survey captures some information about how you overall rate the Sensor-based component of the YAWL system you have been using. Please note again that all information you provide will be treated confidently. Thus, we ask you to please answer honestly.

In the following, you will be given a number of statements on opinions that you may have towards the Sensor-based component of the YAWL system. Please indicate your level of agreement to the statements on the given scale by ticking the box that best describes your view on the respective statement.

Facilitating Conditions	Strongly disagree	Disagree	Somewhat disagree	Normal	Somewhat agree	Agree	Strongly agree
Guidance was available to me in the use of the Sensor-based component of the YAWL system.	1	2	3	4	5	6	7
Specialized instruction concerning the use of the Sensor-based component of the YAWL system was available to me.	1	2	3	4	5	6	7
A specific person or group was available for assistance with difficulties with the Sensor-based component of the YAWL system.	1	2	3	4	5	6	7
Satisfaction	Strongly disagree	Disagree	Somewhat disagree	Normal	Somewhat agree	Agree	Strongly agree
I am extremely pleased with my use of the Sensor-base component of the YAWL system.	1	2	3	4	5	6	7
I am extremely contented with my use of the Sensor-base component of the YAWL system.	1	2	3	4	5	6	7
I am extremely delighted with my use of the Sensor-base component of the YAWL system.	1	2	3	4	5	6	7
I am extremely satisfied with my use of the Sensor-base component of the YAWL system.	1	2	3	4	5	6	7

Perceived Usefulness	Strongly disagree	Disagree	Somewhat disagree	Normal	Somewhat agree	Agree	Strongly agree
Overall, I find the Sensor- based component of the YAWL system useful for modelling process-related risks.	1	2	3	4	5	6	7
I find the Sensor-based component of the YAWL system useful for achieving the purpose of modelling process-related risks.	1	2	3	4	5	6	7
I find the Sensor-based component of the YAWL system helps me in meeting my process-related risks modelling objectives.	1	2	3	4	5	6	7
Perceived Ease of Use	Strongly disagree	Disagree	Somewhat disagree	Normal	Somewhat agree	Agree	Strongly agree
I find it easy to model process- related risks in the way I intended using the Sensor-based component of the YAWL system.	1	2	3	4	5	6	7
I find learning to use the Sensor-based component of the YAWL system is easy.	1	2	3	4	5	6	7
I find easy to create process-related risks using the Sensor-based component of the YAWL system.	1	2	3	4	5	6	7

Intention to Use	Strongly disagree	Disagree	Somewhat disagree	Normal	Somewhat agree	Agree	Strongly agree
I intend to use the Sensor-based component of the YAWL system when I have to define and detect risks in business processes.	1	2	3	4	5	6	7
I predict I would use the Sensor-based component of the YAWL system.	1	2	3	4	5	6	7
I plan to use the Sensor-based component of the YAWL system in the future.	1	2	3	4	5	6	7
I prefer to continue to work with the Sensor-based component of the YAWL system.	1	2	3	4	5	6	7

Bibliography

[10702] 107th Congress USA. Public Law No. 107-204 - Sarbanes-Oxley Act of 2002. July 2002. Cited on pages 1 and 32.

[AAAZ08] W. Steve Albrecht, Conan C. Albrecht, Chad O. Albrecht, and Mark Zimbelman. *Fraud Examination*. South-Western Publishing, 3rd edition, 2008. Cited on page 6.

[AAB+05] Assaf Arkin, Sid Askary, Ben Bloch, Francisco Curbera, Yaron Goland, Neelakantan Kartha, Canyang Kevin Liu, Satish Thatte, Prasad Yendluri, and Alex Yiu. *Web Services Business Process Execution Language Version 2.0*. OASIS, September 2005. Cited on pages 19 and 41.

[Aal11] Wil M. P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011. Cited on pages 13, 50, and 160.

[ABE+05] Birger Andersson, Maria Bergholtz, Ananda Edirisuriya, Tharaka Ilayperuma, and Paul Johannesson. A declarative foundation of process models. In Oscar Pastor and João Falcão e Cunha, editors, *Proceedings of the 17th International Conference Advanced Information Systems Engineering (CAiSE'05), Porto, Portugal, June 13-17, 2005*, volume 3520 of *Lecture Notes in Computer Science*, pages 233–247. Springer, 2005. Cited on pages 28 and 35.

[AD01] Christopher J. Alberts and Audrey J. Dorofee. OCTAVE criteria, version 2.0. Technical Report CMU/SEI-2001-TR-016, Carnegie Mellon University, 2001. Cited on page 24.

[AD02] Wil M. P. van der Aalst and Boudewijn F. van Dongen. Discovering workflow performance models from timed logs. In Yanbo Han, Stefan Tai, and Dietmar Wikarski, editors, *Proceedings of the First*

International Conference on Engineering and Deployment of Cooperative Information Systems (EDCIS'02), Beijing, China, September 17-20, 2002, volume 2480 of *Lecture Notes in Computer Science*, pages 45–63. Springer, 2002. Cited on page 51.

[AG06] Yudistira Asnar and Paolo Giorgini. Modelling risk and identifying countermeasure in organizations. In Javier López, editor, *Proceedings of the First International Workshop on Critical Information Infrastructures Security (CRITIS'06), Samos, Greece, August 31 - September 1, 2006*, volume 4347 of *Lecture Notes in Computer Science*, pages 55–66. Springer, 2006. Cited on page 30.

[AG08] Yudistira Asnar and Paolo Giorgini. Analyzing business continuity through a multi-layers model. In Dumas et al. [DRS08], pages 212–227. Cited on pages 28 and 30.

[AHAE07] Michael Adams, Arthur H. M. ter Hofstede, Wil M. P. van der Aalst, and David Edmond. Dynamic, extensible and context-aware exception handling for workflows. In Robert Meersman and Zahir Tari, editors, *Proceedings Part I of On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS, OTM Confederated International Conferences CoopIS, DOA, ODBASE, GADA, and IS 2007, Vilamoura, Portugal, November 25-30, 2007*, volume 4803 of *Lecture Notes in Computer Science*, pages 95–112. Springer, 2007. Cited on pages 100 and 160.

[AHEA05] Michael Adams, Arthur H. M. ter Hofstede, David Edmond, and Wil M. P. van der Aalst. Facilitating flexibility and dynamic exception handling in workflows through worklets. In Orlando Belo, Johann Eder, João Falcão e Cunha, and Oscar Pastor, editors, *Proceedings of the CAiSE Forum 2005, held in conjunction with the 17th Conference on Advanced Information Systems Engineering (CAiSE'05), Porto, Portugal, 13-17 June, 2005*, volume 161 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005. Cited on pages 4, 45, and 46.

[AHW03] Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and Mathias Weske. Business process management: A survey. In Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and Mathias Weske, editors,

Proceedings of the First International Conference on Business Process Management (BPM'03), Eindhoven, The Netherlands, June 26-27, 2003, volume 2678 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2003. Cited on pages 18, 19, and 53.

[Alt04] Steven L. Alter. A work system view of DSS in its fourth decade. *Decision Support Systems*, 38:319–327, December 2004. Cited on pages 132 and 161.

[ARW+07] Wil M. P. van der Aalst, Hajo A. Reijers, Antonius J. M. M. Weijters, Boudewijn F. van Dongen, Ana Karla Alves de Medeiros, Minseok Song, and H. M. W. (Eric) Verbeek. Business process mining: An industrial application. *Information Systems*, 32(5):713–732, 2007. Cited on page 51.

[AS04] Wil M. P. van der Aalst and Minseok Song. Mining social networks: Uncovering interaction patterns in business processes. In Desel et al. [DPW04], pages 244–260. Cited on page 51.

[ASS11] Wil M. P. van der Aalst, M. Helen Schonenberg, and Minseok Song. Time prediction based on process mining. *Information Systems*, 36(2):450–475, 2011. Cited on pages 52, 71, 90, 113, 132, 134, 139, and 160.

[Bak74] Kenneth R Baker. *Introduction to Sequencing and Scheduling*. Wiley, 1974. Cited on page 130.

[Bas11] Basel Committee on Bankin Supervision. *Basel III: A global regulatory framework for more resilient banks and banking systems*, 2011. Cited on page 1.

[BBK06] Sugato Bagchi, Xue Bai, and Jayant Kalagnanam. Data quality management using business process modeling. In *Proceedings of the 2006 IEEE International Conference on Services Computing (SCC'06), Chicago, Illinois, USA, September 18-22, 2006*, pages 398–405. IEEE Computer Society, 2006. Cited on pages 28 and 33.

[BD92] Barry Barber and John Davey. The use of the CCTA Risk Analysis and Management Methodology CRAMM in health information systems. In *Proceedings of MEDINFO'92*, pages 1589–1593. North Holland Publishing Co, 1992. Cited on page 24.

[BDGG09] Leonora Bianchi, Marco Dorigo, Luca Maria Gambardella, and Walter J. Gutjahr. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, 8:239–287, 2009. Cited on page 63.

[BDL+10] Aliaksandr Birukou, Vincenzo D'Andrea, Frank Leymann, Jacek Serafinski, Patrícia Silveira, Steve Strauch, and Marek Tluczek. An integrated solution for runtime compliance governance in soa. In Paul P. Maglio, Mathias Weske, Jian Yang, and Marcelo Fantinato, editors, *Proceedings of the 8th International Conference on Service-Oriented Computing (ICSOC'10), San Francisco, CA, USA, December 7-10, 2010*, volume 6470 of *Lecture Notes in Computer Science*, pages 122–136, 2010. Cited on page 51.

[Ben86] Jon Bentley. Programming pearls: little languages. *Communications of the ACM*, 29(8):711–721, 1986. Cited on page 76.

[BGJ+05] Maria Bergholtz, Bertrand Grégoire, Paul Johannesson, Michael Schmitt, Petia Wohed, and Jelena Zdravkovic. Integrated methodology for linking business and process models with risk mitigation. In *International Workshop on Requirements Engineering for Business Need and IT Alignment (REBNITA 2005)*. Citeseer, 2005. Cited on pages 28 and 35.

[BHO11] Stefanie Betz, Susan Hickl, and Andreas Oberweis. Risk-Aware Business Process Modeling and Simulation Using XML Nets. In Birgit Hofreiter, Eric Dubois, Kwei-Jay Lin, Thomas Setzer, Claude Godart, Erik Proper, and Lianne Bodenstaff, editors, *Proceedings of the 13th IEEE Conference on Commerce and Enterprise Computing (CEC'11), Luxembourg-Kirchberg, Luxembourg, September 5-7, 2011*, pages 349–356. IEEE, 2011. Cited on pages 28 and 31.

[BIK+07] Moshiur Bhuiyan, M. M. Zahidul Islam, George Koliadis, Aneesh Krishna, and Aditya Ghose. Managing business process risk using rich organizational models. In *Proceedings of the 31st Annual International Computer Software and Applications Conference (COMPSAC'07), Beijing, China, July 24-27, 2007*, pages 509–520. IEEE Computer Society, 2007. Cited on pages 28 and 34.

[BJWW09] Andrew Burton-Jones, Yair Wand, and Ron Weber. Guidelines for

empirical evaluations of conceptual modeling grammars. *Journal of the Association for Information Systems*, 10(6):495–532, 2009. Cited on page 95.

[BKB+14] Nick R. T. P. van Beest, Eirini Kaldeli, Pavel Bulanov, Johan C. Wortmann, and Alexander Lazovik. Automated runtime repair of business processes. *Information Systems*, 39:45–79, 2014. Cited on pages 4 and 46.

[BPK06] Xue Bai, Rema Padman, and Ramayya Krishnan. On risk management in business process design. Technical report, The H. John Heinz III School of Public Policy and Management - Carnegie Mellon University, 2006. Last accessed 15 Sept 2011 from `http://heinz.cmu.edu/research/296full.pdf`. Cited on pages 28 and 33.

[BPK07] Xue Bai, Rema Padman, and Ramayya Krishnan. A risk management approach to business process design. In *Proceedings of the International Conference on Information Systems (ICIS'07), Montreal, Quebec, Canada, December 9-12, 2007*, page 28. Association for Information Systems, 2007. Cited on pages 28 and 33.

[BPZF13] Samik Basu, Cesare Pautasso, Liang Zhang, and Xiang Fu, editors. *Proceedings of the 11th International Conference on Service-Oriented Computing (ICSOC'13), Berlin, Germany, December 2-5, 2013*, volume 8274 of *Lecture Notes in Computer Science*. Springer, 2013. Cited on pages 197 and 218.

[BTWW10] Jörg Becker, Irina Thome, Burkhard Weiß, and Axel Winkelmann. Constructing a semantic business process modelling language for the banking sector - an evolutionary dyadic design science approach. *Enterprise Modelling and Information Systems Architectures*, 5(1):4–25, 2010. Cited on page 30.

[BW04] Philippe Bogaerts and A.Vande Wouwer. Parameter identification for state estimation–application to bioprocess software sensors. *Chemical Engineering Science*, 59(12):2465–2476, 2004. Cited on page 41.

[BWV12] Irene Barba, Barbara Weber, and Carmelo Valle. Supporting the optimized execution of business processes through recommendations.

In La Rosa and Soffer [LS13], pages 135–140. Cited on pages 132 and 133.

[BZ09] Michael Bousamra and Robin Zimmermann. *Oracle BPEL Process Manager 10g Database Schema Partitioning*, May 2009. Cited on page 90.

[CA09] Jorge Cardoso and Wil M. P. van der Aalst. *Handbook of Research on Business Process Modeling*. Handbook of Research On. Information Science Reference, 2009. Cited on pages 48 and 49.

[Cas99] Fabio Casati. A discussion on approaches to handling exceptions in workflows. *ACM SIGGROUP Bulletin*, 20(3):3–4, 1999. Cited on page 100.

[CAS06] Semih Cetin, N. Ilker Altintas, and Remzi Solmaz. Business rules segregation for dynamic process management with an aspect-oriented framework. In Johann Eder and Schahram Dustdar, editors, *Proceedings of the Business Process Management Workshops, BPM 2006 International Workshops, BPD, BPI, ENEI, GPWW, DPM, semantics4ws, Vienna, Austria, September 4-7, 2006*, volume 4103 of *Lecture Notes in Computer Science*, pages 193–204. Springer, 2006. Cited on page 100.

[CCDS04] Malú Castellanos, Fabio Casati, Umeshwar Dayal, and Ming-Chien Shan. A comprehensive and automated approach to intelligent business processes execution analysis. *Distributed and Parallel Databases*, 16(3):239–273, 2004. Cited on pages 49 and 50.

[CDE+10] Eric W. Cope, Lea Deleris, Dominik Etzweiler, Jana Koehler, Jochen M. Kuester, and Bonnie K. Ray. System and method for creating and expressing risk-extended business process models. Patent, 2010. US2010/0179847A1. Cited on pages 28 and 29.

[CDS02] Fabio Casati, Umeshwar Dayal, and Ming-Chien Shan. Business operation intelligence. In Subhash Bhalla, editor, *Proceedings of the Second International Workshop on Databases in Networked Information Systems (DNIS'02), Aizu, Japan, December 16-18, 2002*, volume 2544 of *Lecture Notes in Computer Science*, pages 213–224. Springer, 2002. Cited on pages 49 and 50.

[CFLH11] Raffaele Conforti, Giancarlo Fortino, Marcello La Rosa, and Arthur H. M. ter Hofstede. History-aware, real-time risk detection in business processes. In Meersman et al. [MDH+11], pages 100–118. Cited on page 69.

[CFM99] Fabio Casati, MariaGrazia Fugini, and Isabelle Mirbel. An environment for designing exceptions in workflows. *Information Systems*, 24(3):255–273, 1999. Cited on page 100.

[CGR+13] Cristina Cabanillas, José María García, Manuel Resinas, David Ruiz, Jan Mendling, and Antonio Ruiz Cortés. Priority-based human resource allocation in business processes. In Basu et al. [BPZF13], pages 374–388. Cited on pages 132 and 133.

[Ché97] Arlette Chéruy. Software sensors in bioprocess engineering. *Journal of Biotechnology*, 52(3):193–199, 1997. Cited on page 41.

[CHLA12] Raffaele Conforti, Arthur H. M. ter Hofstede, Marcello La Rosa, and Michael Adams. Automated risk mitigation in business processes. In Meersman et al. [MPD+12], pages 212–231. Cited on page 138.

[CIJ+00] Fabio Casati, Ski Ilnicki, Li-Jie Jin, Vasudev Krishnamoorthy, and Ming-Chien Shan. Adaptive and dynamic service composition in *eflow*. In Benkt Wangler and Lars Bergman, editors, *Proceedings of the 12th International Conference on Advanced Information Systems Engineering (CAiSE'00), Stockholm, Sweden, June 5-9, 2000*, volume 1789 of *Lecture Notes in Computer Science*, pages 13–31. Springer, 2000. Cited on page 100.

[CJ88] Paul Compton and Bob Jansen. Knowledge in context: A strategy for expert system maintenance. In Chris J. Barter and Michael J. Brooks, editors, *Australian Joint Conference on Artificial Intelligence*, volume 406 of *Lecture Notes in Computer Science*, pages 292–306. Springer, 1988. Cited on page 46.

[CJ98] Piotr Czyżżak and Adrezej Jaszkiewicz. Pareto simulated annealing - a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of Multi-Criteria Decision Analysis*, 7(1):34–47, 1998. Cited on page 64.

[CKE09] Eric W. Cope, Jochen Malte Küster, and Dominik Etzweiler. Risk Extensions to the BPMN 1.1 Business Process Metamodel. Technical Report RZ3740, IBM Research, 2009. Cited on pages 28 and 29.

[CKE+10] Eric W. Cope, Jochen Malte Küster, Dominik Etzweiler, Léa Amandine Deleris, and Bonnie Ray. Incorporating risk into business process models. *IBM Journal of Research and Development*, 54(3):4:1–4:13, may-june 2010. Cited on pages 28 and 29.

[CLB04] Paul D. Cousins, Richard C. Lamming, and Frances Bowen. The role of risk in environment-related supplier initiatives. *International Journal of Operations & Production Management*, 24(6):554–565, 2004. Cited on page 85.

[CLF13a] Raffaele Conforti, Marcello La Rosa, and Giancarlo Fortino. Process monitoring using sensors in yawl. In Thomas Freytag, Andreas Hense, Arthur H. M. ter Hofstede, and Jan Mendling, editors, *Proceedings of the First YAWL Symposium, Sankt Augustin, Germany, June 7, 2013*, volume 982 of *CEUR Workshop Proceedings*, pages 49–55. CEUR-WS.org, 2013. Cited on page 69.

[CLF+13b] Raffaele Conforti, Marcello La Rosa, Giancarlo Fortino, Arthur H. M. ter Hofstede, Jan Recker, and Michael J. Adams. Real-Time Risk Monitoring in Business Processes: A Sensor-based Approach. *Journal of Systems and Software*, 86(11):2939–2965, 2013. Cited on pages 69, 125, and 126.

[CLH+13] Raffaele Conforti, Marcello La Rosa, Arthur H. M. ter Hofstede, Giancarlo Fortino, Massimiliano de Leoni, Wil M. P. van der Aalst, and Michael Adams. A software framework for risk-aware business process management. In Rébecca Deneckère and Henderik Alex Proper, editors, *Proceedings of the CAiSE'13 Forum at the 25th International Conference on Advanced Information Systems Engineering (CAiSE'13), Valencia, Spain, June 20th, 2013*, volume 998 of *CEUR Workshop Proceedings*, pages 130–137. CEUR-WS.org, 2013. Cited on pages 69, 105, and 138.

[CLK99] Dickson K. W. Chiu, Qing Li, and Kamalakar Karlapalem. Exception handling with workflow evolution in ADOME-WFMS: a taxon-

omy and resolution techniques. *ACM Siggroup Bulletin*, 20(3):8–8, 1999. Cited on page 100.

[CLL+14] Raffaele Conforti, Massimiliano de Leoni, Marcello La Rosa, Wil M. P. van der Aalst, and Arthur H. M. ter Hofstede. A recommendation system for predicting risks across multiple business process instances. BPM Center Report BPM-14-04, BPMcenter.org, 2014. Cited on page 105.

[CLLA13] Raffaele Conforti, Massimiliano de Leoni, Marcello La Rosa, and Wil M. P. van der Aalst. Supporting risk-informed decisions during business process execution. In Salinesi et al. [SNP13], pages 116–132. Cited on page 105.

[CM07] Anis Charfi and Mira Mezini. AO4BPEL: An aspect-oriented extension to BPEL. *World Wide Web*, 10(3):309–344, 2007. Cited on pages 43, 45, and 160.

[CMZ09] Louise Comfort, Daniel Mosse, and Taieb Znati. Managing risk in real time: Integrating information technology into disaster risk reduction and response. *Commonwealth: A Journal of Political Science*, 15(4):27–45, 2009. Cited on page 67.

[Com90] International Electrotechnical Commission. *IEC 61025 Fault Tree Analysis (FTA)*, 1990. Cited on pages 9, 24, 68, and 70.

[Com95] International Electrotechnical Commission. *IEC 60300-3-9Dependability management - Part 3: Application guide - Section 9: Risk analysis of technological systems- Event Tree Analysis (ETA)*, 1995. Cited on page 24.

[CP09] Carlo Combi and Roberto Posenato. Controllability in temporal conceptual workflow schemata. In Dayal et al. [DEKR09], pages 64–79. Cited on page 159.

[CSC+05] Malú Castellanos, Norman Salazar, Fabio Casati, Umeshwar Dayal, and Ming-Chien Shan. Predictive business operations management. In Subhash Bhalla, editor, *Proceedings of the 4th International Workshop on Databases in Networked Information Systems (DNIS'05), Aizu-Wakamatsu, Japan, March 28-30, 2005*, volume

3433 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2005. Cited on page 52.

[CSI11] Ran Cheng, Shazia Wasim Sadiq, and Marta Indulska. Framework for business process and rule integration: A case of bpmn and sbvr. In Witold Abramowicz, editor, *Proceedings of the 14th International Conference on Business Information Systems (BIS'11), Poznan, Poland, June 15-17, 2011*, volume 87 of *Lecture Notes in Business Information Processing*, pages 13–24. Springer, 2011. Cited on page 34.

[DAH05] Marlon Dumas, Wil M. P. van der Aalst, and Arthur H. M. ter Hofstede. *Process-Aware Information Systems: Bridging People and Software through Process Technology*. Wiley & Sons, 2005. Cited on pages 1, 4, and 9.

[Dan51] George B. Dantzig. Maximization of a linear function subject to linear inequalities. In Tjalling C. Koopmans, editor, *Activity analysis of production and allocation*, pages 339–347. John Wiley & Sons, 1951. Cited on page 62.

[Dav89] Fred D. Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly*, 13:319–340, 1989. Cited on page 98.

[DB07] Rob B. Davis and Eric Brabander. *ARIS Design Platform: Getting Started with BPM*. Springer, 2007. Cited on pages 19, 42, and 101.

[DBL10] *Proceedings of the Fifth International Conference on Availability, Reliability and Security (ARES'10), Krakow, Poland, February 15-18, 2010*. IEEE Computer Society, 2010. Cited on pages 208 and 224.

[DBM88] Umeshwar Dayal, Alejandro P. Buchmann, and Dennis R. McCarthy. Rules are objects too: A knowledge model for an active, object-oriented database system. In Klaus R. Dittrich, editor, *Advances in Object-Oriented Database Systems, Proceedings of the 2nd International Workshop on Object-Oriented Database Systems, Bad Münster am Stein-Ebernburg, FRG, September 27-30, 1988*, volume 334 of *Lecture Notes in Computer Science*, pages 129–143. Springer, 1988. Cited on pages 38 and 100.

[DEKR09] Umeshwar Dayal, Johann Eder, Jana Koehler, and Hajo A. Reijers, editors. *Proceedings of the 7th International Conference on Business Process Management (BPM'09), Ulm, Germany, September 8-10, 2009*, volume 5701 of *Lecture Notes in Computer Science*. Springer, 2009. Cited on pages 199 and 202.

[DLMR13] Marlon Dumas, Marcello La Rosa, Jan Mendling, and Hajo A. Reijers. *Fundamentals of Business Process Management*. Springer, 2013. Cited on pages 17, 18, and 21.

[DPW04] Jörg Desel, Barbara Pernici, and Mathias Weske, editors. *Proceedings of the Second International Conference on Business Process Management (BPM'04), Potsdam, Germany, June 17-18, 2004*, volume 3080 of *Lecture Notes in Computer Science*. Springer, 2004. Cited on pages 193 and 218.

[DR09] Peter Dadam and Manfred Reichert. The ADEPT project: a decade of research and development for robust and flexible process support. *Computer Science - R&D*, 23(2):81–97, 2009. Cited on pages 44 and 159.

[DRS08] Marlon Dumas, Manfred Reichert, and Ming-Chien Shan, editors. *Proceedings of the 6th International Conference on Business Process Management (BPM'08), Milan, Italy, September 2-4, 2008*, volume 5240 of *Lecture Notes in Computer Science*. Springer, 2008. Cited on pages 192 and 223.

[EFKW07] Andreas Ekelhart, Stefan Fenz, Markus D. Klemen, and Edgar Weippl. Security ontologies: Improving quantitative risk analysis. In *Proceedings of the 40th Hawaii International International Conference on Systems Science (HICSS'07), Waikoloa, Big Island, HI, USA, January 3-6, 2007*, pages 156–162. IEEE Computer Society, 2007. Cited on pages 27 and 28.

[Elm93] Salah E. Elmaghraby. Resource allocation via dynamic programming in activity networks. *European Journal of Operational Research*, 64(2):199 – 215, 1993. Cited on page 130.

[FB13] Frances Faulds and Joel Bessis. Rogue trading: Back to front. *Journal of Risk Management in Financial Institutions*, 6(1):4–5, 2013. Cited on page 1.

[FE09] Stefan Fenz and Andreas Ekelhart. Formalizing information security knowledge. In Wanqing Li, Willy Susilo, Udaya Kiran Tupakula, Reihaneh Safavi-Naini, and Vijay Varadharajan, editors, *Proceedings of the 2009 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2009, Sydney, Australia, March 10-12, 2009*, pages 183–194. ACM, 2009. Cited on pages 28 and 34.

[FEN09] Stefan Fenz, Andreas Ekelhart, and Thomas Neubauer. Business process-based resource importance determination. In Dayal et al. [DEKR09], pages 113–127. Cited on pages 28 and 34.

[Fen10] Stefan Fenz. From the resource to the business process risk level. In Nathan L. Clarke, Steven Furnell, and Rossouw von Solms, editors, *Proceedings South African Information Security Multi-Conference (SAISMC'10), Port Elizabeth, South Africa, May 17-18, 2010*, pages 100–109. University of Plymouth, 2010. Cited on pages 28 and 34.

[FGP12] Francesco Folino, Massimo Guarascio, and Luigi Pontieri. Discovering context-aware models for predicting business process performances. In Meersman et al. [MPD+12], pages 287–304. Cited on pages 52, 132, and 134.

[FN09] Stefan Fenz and Thomas Neubauer. How to determine threat probabilities using ontologies and bayesian networks. In Frederick T. Sheldon, Greg Peterson, Axel W. Krings, Robert K. Abercrombie, and Ali Mili, editors, *Proceedings of the Fifth Cyber Security and Information Intelligence Research Workshop (CSIIRW'09), Knoxville, TN, USA, April 13-15, 2009*, page 69. ACM, 2009. Cited on pages 28 and 34.

[Fra10] Ulrich Frank. *The MEMO meta modelling language (MML) and language architecture*. Institute for Computer Science and Business Information Systems (ICB), 2010. Cited on page 32.

[GAC+08] Willie Golden, Thomas Acton, Kieran Conboy, Hans van der Heijden, and Virpi Kristiina Tuunainen, editors. *Proceedings of the 16th European Conference on Information Systems (ECIS'08), Galway, Ireland, 2008*, 2008. Cited on pages 207 and 219.

[Gar14] EXP Gartner. How the pieces in a bam architecture work. *Gartner Inc, Stamford, Connecticut*, `http://www.gartner.com/newsroom/id/2304615`. Accessed: April 2014. Cited on pages 1 and 7.

[Gas04] Bill Gassman. How the pieces in a bam architecture work. Technical report, Gartner Inc, Stamford, Connecticut, `https://www.gartner.com/doc/430320/pieces-bam-architecture-work`. April 2004. Cited on page 40.

[GCC+04] Daniela Grigori, Fabio Casati, Malú Castellanos, Umeshwar Dayal, Mehmet Sayal, and Ming-Chien Shan. Business process intelligence. *Computers in Industry*, 53(3):321–343, 2004. Cited on pages 49 and 50.

[GEF+08] Gernot Goluch, Andreas Ekelhart, Stefan Fenz, Stefan Jakoubi, Simon Tjoa, and Thomas Mück. Integration of an ontological information security concept in risk aware business process management. In *Proceedings of the 41st Hawaii International International Conference on Systems Science (HICSS-41 2008), Waikoloa, Big Island, HI, USA, January 7-10, 2008*, pages 377–395. IEEE Computer Society, 2008. Cited on pages 27 and 28.

[GG84] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-6(6):721–741, Nov 1984. Cited on page 64.

[GG85] Benny Gilad and Tamar Gilad. A systems approach to business intelligence. *Business Horizons*, 28(5):65–70, Sept.-Oct 1985. Cited on page 48.

[GLMH11] Mauro Gambini, Marcello La Rosa, Sara Migliorini, and Arthur H. M. ter Hofstede. Automated error correction of business process models. In Rinderle-Ma et al. [RMTW11], pages 148–165. Cited on page 161.

[Gom58] Ralph E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64:275–278, 1958. Cited on page 62.

[GPL+10] Pablo Gay, Albert Pla, Beatriz López, Joaquím Meléndez, and Regine Meunier. Service workflow monitoring through complex event processing. In *Proceedings of 15th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'10), Bilbao, Spain, September 13-16, 2010*, pages 1–4. IEEE, 2010. Cited on pages 6, 42, 43, and 101.

[GRRC10] JosĂ©MarĂa GarcĂa, David Ruiz, and Antonio Ruiz-CortĂ©s. A model of user preferences for semantic services discovery and ranking. In Lora Aroyo, Grigoris Antoniou, Eero HyvĂ¶nen, Annette ten Teije, Heiner Stuckenschmidt, Liliana Cabral, and Tania Tudorache, editors, *The Semantic Web: Research and Applications*, volume 6089 of *LNCS*, pages 1–14. Springer, 2010. Cited on page 133.

[GV13] Christian W. Günther and Eric Verbeek. Openxes developer guide version 1.9. Technical Report 55979, Eindhoven University of Technology, `http://www.xes-standard.org/_media/openxes/\openxesdeveloperguide-1.9.pdf`, 2013. Cited on page 75.

[GYR06] Jaap Gordijn, Eric Yu, and Bas van der Raadt. E-service design using i^* and e^3-value modeling. *Software, IEEE*, 23(3):26–33, 2006. Cited on page 35.

[HA00] Claus Hagen and Gustavo Alonso. Exception handling in workflow management systems. *Software Engineering, IEEE Transactions on*, 26(10):943–958, Oct 2000. Cited on pages 45 and 46.

[HAAR10] Arthur H. M. ter Hofstede, Wil M. P. van der Aalst, Michael J. Adams, and Nick Russell, editors. *Modern Business Process Automation: YAWL and its Support Environment*. Springer, 2010. Cited on pages 4, 12, 18, 19, 55, 56, 57, 68, and 152.

[HB99] Terry Halpin and Anthony Bloesch. Data modeling in UML and ORM: a comparison. *Journal of Database Management*, 10(4):4–13, 1999. Cited on page 76.

[HBW03] Christine Harland, Richard Brenchley, and Helen Walker. Risk in supply networks. *Journal of Purchasing and Supply Management*, 9(2):51–62, 2003. Cited on page 85.

[HH06] Peter Herrmann and Gaby Herrmann. Security requirement analysis of business processes. *Electronic Commerce Research*, 6(3-4):305–335, 2006. Cited on pages 28 and 32.

[Hig] High Court of Australia. Patel v The Queen [2012] HCA 29 (24 August 2012). `http://www.austlii.edu.au/au/cases/cth/HCA/2012/29.html` Accessed 2 May 2013. Cited on page 1.

[HLD11] Zhengxing Huang, Xudong Lu, and Huilong Duan. Mining association rules to support resource allocation in business process management. *Expert Systems with Applications*, 38(8):9483–9490, 2011. Cited on page 134.

[HLD12] Zhemgxomg Huang, Xudong Lu, and Huilong Duan. A task operation model for resource allocation optimization in business process management. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 42(5):1256–1270, 2012. Cited on pages 132 and 134.

[HMPR04] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75–105, 2004. Cited on page 8.

[Hol95] David Hollingsworth. *The Workflow Reference Model. Workflow Management Coalition.* Workflow Management Coalition, 1995. Cited on pages 22 and 53.

[Hop10] Paul Hopkin. *Fundamentals of Risk Management: Understanding, Evaluating and Implementing Effective Risk Management.* Kogan Page Series. Kogan Page, 2010. Cited on page 5.

[HS65] Richard F. Hespos and Paul A. Strassmann. Stochastic Decision Trees for the Analysis of Investment Decisions. *Management Science*, 11(10):244–259, 1965. Cited on page 6.

[HSD10] Gabriel Hermosillo, Lionel Seinturier, and Laurence Duchien. Using complex event processing for dynamic business process adaptation. In *Proceedings of the 2010 IEEE International Conference on Services Computing (SCC'10), Miami, Florida, USA, July 5-10, 2010*, pages 466–473. IEEE Computer Society, 2010. Cited on pages 6, 42, 43, 44, 101, and 160.

[IBKG09] M. M. Zahidul Islam, Moshiur Bhuiyan, Aneesh Krishna, and Aditya Ghose. An integrated approach to managing business process risk using rich organizational models. In Robert Meersman, Tharam S. Dillon, and Pilar Herrero, editors, *Proceedings Part I of On the Move to Meaningful Internet Systems: OTM 2009, Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009, Vilamoura, Portugal, November 1-6, 2009*, volume 5870 of *Lecture Notes in Computer Science*, pages 273–285. Springer, 2009. Cited on pages 28 and 34.

[IDE93] Integration Definition for Function Modeling (IDEF0). *Federal Information Processing Standards*, 183, 1993. Cited on page 33.

[Int01] International Electrotechnical Commission. *IEC 61882 Hazard and operability studies (HAZOP studies) - Application guide*, 2001. Cited on page 24.

[Int09] International Organization for Standardization. *ISO Guide 73 Risk management - Vocabulary*, 2009. Cited on pages 2 and 22.

[Jac10] Laurent L. Jacque. *Global Derivative Debacles: From Theory to Malpractice*, chapter 11: Société Générale, pages 179–196. World Scientific, 2010. Cited on page 1.

[JDV11] Mieke Jans, Benoît Depaire, and Koen Vanhoof. Does process mining add to internal auditing? an experience report. In Terry Halpin, Selmin Nurcan, John Krogstie, Pnina Soffer, Erik Proper, Rainer Schmidt, and Ilia Bider, editors, *Proceedings of the Twelfth International Conference on Business Process Modeling, Development, and Support (BPMDS'11), and the Sixteenth International Conference on Exploring Modelling Methods for Systems Analysis and Design (EMMSAD'11), held in conjunction with The Twenty-third International Conference on Advanced Information Systems Engineering (CAiSE'11), London, UK, June 20-21, 2011*, volume 81 of *Lecture Notes in Business Information Processing*, pages 31–45. Springer, 2011. Cited on pages 28 and 36.

[JGTQ08] Stefan Jakoubi, Gernot Goluch, Simon Tjoa, and Gerald Quirchmayr. Deriving resource requirements applying risk-aware business

process modeling and simulation. In Golden et al. [GAC+08], pages 1542–1554. Cited on pages 27 and 28.

[JLV01] Mieke Jans, Nadine Lybaert, and Koen Vanhoof. Business process mining for internal fraud risk reduction: Results of a case study. In *Proceedings of the International Research Symposium on Accounting Information Systems, (IRSAIS'08)*, 2001. `http://doclib.uhasselt.be/dspace/bitstream/1942/10457/1/Jansetal-paper.pdf` (current April 15, 2013). Cited on pages 28 and 36.

[JMV+07] Abdou Karim Jallow, Basim Majeed, Kostas Vergidis, Ashutosh Tiwari, and Rajkumar Roy. Operational risk analysis in business processes. *BT Technology Journal*, 25(1):168–177, 2007. Cited on pages 28 and 35.

[JNT09] Stefan Jakoubi, Thoma Neubauer, and Simon Tjoa. A roadmap to risk-aware business process management. In Markus Kirchberg, Patrick C. K. Hung, Barbara Carminati, Chi-Hung Chi, Rajaraman Kanagasabai, Emanuele Della Valle, Kun-Chan Lan, and Ling-Jyh Chen, editors, *Proceedings of the 4th IEEE Asia-Pacific Services Computing Conference (IEEE APSCC'09), Singapore, December 7-11, 2009*, pages 23 –27. IEEE, 2009. Cited on pages 27 and 28.

[JO98] Martin James and James J. Odell. *Object-Oriented Methods: a Foundaton - UML Edition*. Prentice Hall PTR, Upper Saddle River, New Jersey, USA, 1998. Cited on page 38.

[Joh73] William G. Johnson. *MORT - The Management Oversight and Risk Tree*. U.S. Atomic Energy Commission, 1973. Cited on page 9.

[JT09] Stefan Jakoubi and Simon Tjoa. A reference model for risk-aware business process management. In Anas Abou El Kalam, Yves Deswarte, and Mahmoud Mostafa, editors, *Proceedings of the Fourth International Conference on Risks and Security of Internet and Systems (CRiSIS'09), Toulouse, France, October 19-22, 2009*, pages 82–89. IEEE, 2009. Cited on pages 27, 28, and 29.

[JTGK10a] Stefan Jakoubi, Simon Tjoa, Sigrun Goluch, and Gerhard Kitzler. A formal approach towards risk-aware service level analysis and planning. In *Proceedings of the Fifth International Conference on*

Availability, Reliability and Security (ARES'10), Krakow, Poland, February 15-18, 2010 [DBL10], pages 180–187. Cited on pages 27 and 28.

[JTGK10b] Stefan Jakoubi, Simon Tjoa, Sigrun Goluch, and Gerhard Kitzler. Risk-aware business process management: Establishing the link between business and security. In *Complex Intelligent Systems and Their Applications*, volume 41 of *Optimization and its Applications*, pages 109–135. Springer, 2010. Cited on pages 27, 28, and 29.

[JTQ07] Stefan Jakoubi, Simon Tjoa, and Gerlad Quirchmayr. ROPE: A methodology for enabling the risk-aware modelling and simulation of business processes. In Hubert Österle, Joachim Schelp, and Robert Winter, editors, *Proceedings of the Fifteenth European Conference on Information Systems (ECIS'07), St. Gallen, Switzerland, 2007*. University of St. Gallen, 2007. Cited on pages 27 and 28.

[JWLV11] Mieke Jans, Jan Martijn van der Werf, Nadine Lybaert, and Koen Vanhoof. A business process mining application for internal transaction fraud mitigation. *Expert Systems with Applications*, 38:13351–13359, 2011. Cited on pages 28 and 36.

[KAA+11] Thorsten Koch, Tobias Achterberg, Erling Andersen, Oliver Bastert, Timo Berthold, RobertE. Bixby, Emilie Danna, Gerald Gamrath, AmbrosM. Gleixner, Stefan Heinz, Andrea Lodi, Hans Mittelmann, Ted Ralphs, Domenico Salvagnin, DanielE. Steffy, and Kati Wolter. Miplib 2010. *Mathematical Programming Computation*, 3(2):103–163, 2011. Cited on page 127.

[KAV02] Akhil Kumar, Wil M. P. van der Aalst, and Eric M. W. Verbeek. Dynamic work distribution in workflow management systems: How to balance quality and performance. *Journal of Management Information Systems*, 18(3):157–193, January 2002. Cited on pages 131, 132, and 133.

[KCK09] Bokyoung Kang, Nam Wook Cho, and Suk-Ho Kang. Real-time risk measurement for business activity monitoring (BAM). *International Journal of Innovative Computing, Information and Control*, 5(11):3647–3657, 2009. Cited on pages 6, 28, 36, and 53.

[KDS13] Akhil Kumar, Remco M. Dijkman, and Minseok Song. Optimal resource assignment in workflows for maximizing cooperation. In Florian Daniel, Jianmin Wang, and Barbara Weber, editors, *BPM*, volume 8094 of *Lecture Notes in Computer Science*, pages 235–250. Springer, 2013. Cited on pages 132 and 134.

[KEP00] Gerhard Knolmayer, Rainer Endl, and Marcel Pfahrer. Modeling processes and workflows by business rules. In Wil M. P. van der Aalst, Jörg Desel, and Andreas Oberweis, editors, *Business Process Management, Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2000. Cited on page 38.

[KGV83] Scott Kirkpatrick, C. Daniel Gelatt, and Mario P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983. Cited on page 64.

[KKK07] Dongsoo Kim, Minsoo Kim, and Hoontae Kim. Dynamic business process management based on process change patterns. In *Convergence Information Technology, 2007. International Conference on*, pages 1154 –1161, Nov 2007. Cited on pages 44, 45, and 46.

[KMS07] Dimitris Karagiannis, John Mylopoulos, and Margit Schwab. Business process-based regulation compliance: The case of the sarbanes-oxley act. In *Proceedings of the 15th IEEE International Requirements Engineering Conference (RE'07), New Delhi, India, October 15-19, 2007*, pages 315–321. IEEE, 2007. Cited on pages 28 and 32.

[KMZN06] Manuel Kaegi, Ralf Mock, Ruth Ziegler, and Roberto Nibali. Information systems' risk analysis by agent-based modelling of business processes. In Carlos Guedes Soares and Enrico Zio, editors, *Safety and Reliability for Managing Risk - Proceedings of the 15th European Safety and Reliability Conference (ESREL 2006), Estoril, Portugal, September 18-22, 2006*, volume 3, pages 2277–2284. Taylor & Francis, 2006. Cited on pages 28 and 35.

[KOJ14] Aekyung Kim, Josue Obregon, and Jae-Yoon Jung. Constructing decision trees from process logs for performer recommendation. In *Proceedings of the First International Workshop on Decision Mining & Modeling for Business Processes (DeMiMoP'13), held in con-*

junction with the 11th International Conference on Business Process Management (BPM'13), Beijing, China, August 26, 2013, Lecture Notes in Business Information Processing. Springer, 2014. To appear. Cited on pages 132 and 134.

[KSLP07] Achim P. Karduck, Amadou Sienou, Elyes Lamine, and Hervé Pingaud. Collaborative process driven risk management for enterprise agility. In *Proceedings of the First IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST'07), Cairns, Australia, February 21-23, 2008*, pages 472–477. IEEE, 2007. Cited on pages 28 and 29.

[KT98] Gerhard Keller and Thomas Teufel. *SAP R/3 Process Oriented Implementation: Iterative Process Prototyping*. Addison-Wesley Longman Publishing Co., Inc, 1998. Cited on page 31.

[LAAH12] Massimiliano de Leoni, Michael J. Adams, Wil M. P. van der Aalst, and Arthur H. M. ter Hofstede. Visual support for work assignment in process-aware information systems: Framework formalisation and implementation. *Decision Support Systems*, 54(1):345–361, 2012. Cited on pages 120, 121, and 132.

[LD60] Ailsa H. Land and Alison G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28:497–520, 1960. Cited on page 62.

[LHW+11] Marcello La Rosa, Arthur H. M. ter Hofstede, Petia Wohed, Hajo A. Reijers, Jan Mendling, and Wil M. P. van der Aalst. Managing process model complexity via concrete syntax modifications. *Industrial Informatics, IEEE Transactions on*, 7(2):255–265, 2011. Cited on page 94.

[LJJ06] James H. Lambert, Rachel K. Jennings, and Nilesh N. Joshi. Integration of risk identification with business process models. *Systems engineering*, 9(3):187–198, 2006. Cited on pages 28 and 33.

[LO03] Kirsten Lenz and Andreas Oberweis. Inter-organizational Business Process Management with XML Nets. In Hartmut Ehrig, Wolfgang Reisig, Grzegorz Rozenberg, and Herbert Weber, editors, *Petri Net Technology for Communication-Based Systems - Advances in Petri*

Nets, volume 2472 of *Lecture Notes in Computer Science*, pages 243–263. Springer, 2003. Cited on page 31.

[LRMKD11] Linh Thao Ly, Stefanie Rinderle-Ma, David Knuplesch, and Peter Dadam. Monitoring business process compliance using compliance rule graphs. In Meersman et al. [MDH+11], pages 82–99. Cited on page 51.

[LS13] Marcello La Rosa and Nina Soffer, editors. *Proceedings of the Business Process Management Workshops, held in conjunction with the 10th International on Business Process Management (BPM'12), Tallinn, Estonia, September 3, 2012*, volume 132 of *Lecture Notes in Business Information Processing*. Springer, 2013. Cited on pages 196, 215, and 222.

[LSKA03] Zongwei Luo, Amit Sheth, Krys Kochut, and Budak Arpinar. Exception handling for conflict resolution in cross-organisational workflows. *Distributed and Parallel Databases*, 11:271–306, May 2003. Cited on page 100.

[LSS11] Mass Soldal Lund, Bjørnar Solhaug, and Ketil Stølen. *Model-Driven Risk Analysis - The CORAS Approach*. Springer, 2011. Cited on pages 5, 9, 23, and 24.

[LWM+11] Marcello La Rosa, Petia Wohed, Jan Mendling, Arthur H. M. ter Hofstede, Hajo A. Reijers, and Wil M. P. van der Aalst. Managing process model complexity via abstract syntax modifications. *Industrial Informatics, IEEE Transactions on*, 7(4):614–629, 2011. Cited on page 94.

[LWYS08] Yingbo Liu, Jianmin Wang, Yun Yang, and Jiaguang Sun. A semi-automatic approach for workflow staff assignment. *Computers in Industry*, 59(5):463–476, 2008. Cited on page 134.

[MC05] Ralf Mock and Maurizio Corvo. Risk analysis of information systems by event process chains. *International Journal of Critical Infrastructures*, 1(2-3):247–257, 2005. Cited on pages 28 and 30.

[MDDG14] Fabrizio Maria Maggi, Chiara Di Francescomarino, Marlon Dumas, and Chiara Ghidini. Predictive monitoring of business processes.

In Matthias Jarke, John Mylopoulos, Christoph Quix, Colette Rolland, Yannis Manolopoulos, Haralambos Mouratidis, and Jennifer Horkoff, editors, *Proceedings of the 26th International Conference on Advanced Information Systems Engineering (CAiSE'14), Thessaloniki, Greece, June 16-20, 2014*, volume 8484 of *Lecture Notes in Computer Science*, pages 457–472. Springer, 2014. Cited on pages 52, 131, 132, and 134.

[MDH+11] Robert Meersman, Tharam S. Dillon, Pilar Herrero, Akhil Kumar, Manfred Reichert, Li Qing, Beng Chin Ooi, Ernesto Damiani, Douglas C. Schmidt, Jules White, Manfred Hauswirth, Pascal Hitzler, and Mukesh K. Mohania, editors. *Proceedings Part I of the On the Move to Meaningful Internet Systems: OTM 2011 - Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2011, Hersonissos, Crete, Greece, October 17-21, 2011*, volume 7044 of *Lecture Notes in Computer Science*. Springer, 2011. Cited on pages 197 and 211.

[Met87] Nicholas C. Metropolis. The beginning of the monte carlo method. *Los Alamos Science*, 15(584):125–130, 1987. Cited on page 35.

[Meu00] Lisa Meulbroek. Total strategies for company-wide risk control. *Financial Times*, 9:1–4, 2000. Cited on page 85.

[Mey90] Bertrand Meyer. *Introduction to the theory of programming languages*. Prentice-Hall, 1990. Cited on page 76.

[MGR04] Robert Müller, Ulrike Greiner, and Erhard Rahm. AgentWork: a workflow system supporting rule-based workflow adaptation. *Data & Knowledge Engineering*, 51(2):223–256, 2004. Cited on pages 44 and 160.

[MH+04] Deborah L. McGuinness, Frank van Harmelen, et al. OWL Web Ontology Language Overview. *W3C recommendation*, 10, 2004. Cited on page 34.

[MH05] Michael zur Muehlen and Danny Ting-Yi Ho. Risk Management in the BPM Lifecycle. In Christoph Bussler and Armin Haller, editors, *Proceedings of the Business Process Management Workshops, BPM 2005 International Workshops, BPI, BPD, ENEI, BPRM, WSCOBPM, BPS, Nancy, France, September 5, 2005*, volume 3812

of *Lecture Notes in Computer Science*, pages 454–466, 2005. Cited on pages 6 and 9.

[MMA12] Fabrizio Maria Maggi, Marco Montali, and Wil M. P. van der Aalst. An operational decision support framework for monitoring business constraints. In Juan de Lara and Andrea Zisman, editors, *Proceedings of the 15th International Conference on Fundamental Approaches to Software Engineering (FASE'12), held as part of the European Joint Conferences on Theory and Practice of Software, (ETAPS'12), Tallinn, Estonia, March 24 - April 1, 2012*, volume 7212 of *Lecture Notes in Computer Science*, pages 146–162. Springer, 2012. Cited on page 51.

[MMC+13] Marco Montali, Fabrizio Maria Maggi, Federico Chesani, Paola Mello, and Wil M. P. van der Aalst. Monitoring business constraints with the event calculus. *ACM Transactions on Intelligent Systems and Technology*, 5(1):17:1–17:30, 2013. Cited on page 131.

[MMWA11] Fabrizio Maria Maggi, Marco Montali, Michael Westergaard, and Wil M. P. van der Aalst. Monitoring business constraints with linear temporal logic: An approach based on colored automata. In Rinderle-Ma et al. [RMTW11], pages 132–147. Cited on page 51.

[MPD+12] Robert Meersman, Hervé Panetto, Tharam S. Dillon, Stefanie Rinderle-Ma, Peter Dadam, Xiaofang Zhou, Siani Pearson, Alois Ferscha, Sonia Bergamaschi, and Isabel F. Cruz, editors. *Proceedings Part I of On the Move to Meaningful Internet Systems: OTM 2012, Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2012, Rome, Italy, September 10-14, 2012*, volume 7565 of *Lecture Notes in Computer Science*. Springer, 2012. Cited on pages 197, 202, and 213.

[MRM12] Andrea Marrella, Alessandro Russo, and Massimo Mecella. Planlets: Automatically recovering dynamic processes in yawl. In Meersman et al. [MPD+12], pages 268–286. Cited on pages 4 and 46.

[MRR+53] Nicholas C. Metropolis, Arianna W. Rosenbluth., Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087, 1953. Cited on page 64.

[MSR12] Jan Mendling, Mark Strembeck, and Jan Recker. Factors of process model comprehension - findings from a series of experiments. *Decision Support Systems*, 53(1):195–206, 2012. Cited on page 95.

[MZW12] Fatemeh Hoda Mogihim, Hossein Zadeh, and Nilmini Wickramasinghe. An intelligence e-risk detection model to improve decision efficiency in the context of the orthopaedic operating room. In *Critical Issues for the Development of Sustainable E-health Solutions*, pages 17–32. Springer, 2012. Cited on page 67.

[NCMR06] Dina Neiger, Leonid Churilov, Michael zur Muehlen, and Michael Rosemann. Integrating risks in business process models with value focused process engineering. In Jan Ljungberg and Magnus Andersson, editors, *Proceedings of the Fourteenth European Conference on Information Systems (ECIS'06), Göteborg, Sweden, 2006*, pages 1606–1615, 2006. Cited on pages 9, 28, and 31.

[Net10] Mariska Netjes. *Process Improvement: the Creation and Evaluation of Process Alternatives*. PhD thesis, Eindhoven University of Technology, 2010. Cited on pages 47, 48, and 53.

[Nie71] Dan S. Nielsen. The cause/consequence diagram method as basis for quantitative accident analysis. Technical report RISO-M-1374, Danish Atomic Energy Commission, 1971. Cited on page 24.

[NWA12] Joyce Nakatumba, Michael Westergaard, and Wil M. P. van der Aalst. A meta-model for operational support. BPM Center Report BPM-12-05, BPMcenter.org, 2012. Cited on page 131.

[Obj08a] Object Management Group (OMG). *Business Process Model and Notation (BPMN) ver. 1.1*, January 2008. Cited on pages 19 and 29.

[Obj08b] Object Management Group (OMG). *Semantics of Business Vocabulary and Business Rules (SBVR), Version 1.0*, Jan 2008. Cited on page 39.

[Obj09a] Object Management Group, Inc. (OMG). *OMG Unified Modeling Language (OMG UML), Infrastructure, Version 2.2*, Feb 2009. Cited on page 19.

[Obj09b] Object Management Group, Inc. (OMG). *OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.2*, Feb 2009. Cited on page 19.

[Obj11] Object Management Group (OMG). *Business Process Model and Notation (BPMN) ver. 2.0*. Object Management Group (OMG), January 2011. Cited on pages 4 and 73.

[OLH+08] Chun Ouyang, Marcello La Rosa, Arthur H. M. ter Hofstede, Marlon Dumas, and Katherine Shortland. Toward web-scale workflows for film production. *IEEE Internet Computing*, 12(5):53–61, 2008. Cited on pages 138 and 153.

[ONS96] T. B. Önnerth, M. K. Nielsen, and C. Stamer. Advanced computer controlbased on real and software sensors. *Water Science and Technology*, 33(1):237–245, 1996. Sensors in Wastewater Technology, Selected Proceedings of the IAWQ Specialized Conference on Sensors in Wastewater Technology. Cited on page 41.

[Ora11] Oracle. *BPEL Process Manager Developer's Guide*, `http://download.oracle.com/docs/cd/E15523_01/integration.1111/e10224/bp_sensors.htm`. Accesssed: Jun. 2011. Cited on pages 6 and 40.

[Ost04] Alexander Osterwalder. *The Business Model Ontology*. PhD thesis, Universite de Lausanne - Ecole des Hautes Etudes Commerciales, 2004. Cited on page 35.

[PAF+13a] Anastasiia Pika, Wil M. P. van der Aalst, Colin Fidge, Arthur H. M. ter Hofstede, and Moe Wynn. Predicting deadline transgressions using event logs. In La Rosa and Soffer [LS13], pages 211–216. Cited on pages 28 and 37.

[PAF+13b] Anastasiia Pika, Wil M. P. van der Aalst, Colin J. Fidge, Arthur H. M. ter Hofstede, and Moe Thandar Wynn. Profiling event logs to configure risk indicators for process delays. In Salinesi et al. [SNP13], pages 465–481. Cited on pages 28 and 37.

[PH03] Virpi Pirttimäki and Mika Hannula. Process Models of Business Intelligence. In M. Hannula, A.-M Järvelin, and M. Seppä, editors,

Frontiers of e-Business Research 2003, pages 250–260, Tampere, 2003. Cited on page 48.

[POAG10] Nikolaos A. Panayiotou, Stylianos Oikonomitsios, Christina Athanasiadou, and Sotiris P. Gayialis. Risk assessment in virtual enterprise networks: A process-driven internal audit approach. In *Managing Risk in Virtual Enterprise Networks: Implementing Supply Chain Principles*. IGI Global, 2010. Cited on pages 28 and 33.

[PR91] Manfred Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *Siam Review*, 33:60–100, 1991. Cited on page 62.

[PSA07] Maja Pesic, M. Helen Schonenberg, and Wil M. P. van der Aalst. DECLARE: Full Support for Loosely-Structured Processes. In *Proceedingd of the 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC'07), Annapolis, Maryland, USA, October 15-19, 2007*, pages 287–300. IEEE Computer Society, 2007. Cited on page 39.

[PSA09] Maja Pesic, M. Helen Schonenberg, and Wil M. P. van der Aalst. DECLARE Demo: A Constraint-based Workflow Management System. In Ana Karla A. de Medeiros and Barbara Weber, editors, *Proceedings of the Business Process Management Demonstration Track (BPMDemos 2009), Ulm, Germany, September 8, 2009*, volume 489 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009. Cited on page 39.

[Qui93] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc, 1993. Cited on page 110.

[RA05] Hajo A. Reijers and Wil M.P. van der Aalst. The effectiveness of workflow management systems: Predictions and lessons learned. *International Journal of Information Management*, 25(5):458–472, 2005. Cited on page 1.

[RAH06] Nick Russell, Wil M. P. van der Aalst, and Arthur H. M. ter Hofstede. Workflow exception patterns. In Eric Dubois and Klaus Pohl, editors, *Proceedings of the 18th International Conference on Advanced Information Systems Engineering (CAiSE'06), Luxembourg,*

Luxembourg, June 5-9, 2006, volume 4001 of *Lecture Notes in Computer Science*, pages 288–302. Springer, 2006. Cited on pages 158 and 159.

[RBGR06] Pall Rikhardsson, Peter J Best, Peter Green, and Michael Rosemann. Business process risk management and internal control: A proposed research agenda in the context of compliance and ERP systems. In *Proceedings of the Second Asia/Pacific Research Symposium on Accounting Information Systems*, Melbourne, Australia, 2006. Cited on page 2.

[RD98] Manfred Reichert and Peter Dadam. Adept$_{flex}$ - Supporting Dynamic Changes of Workflows Without Losing Control. *Journal of Intelligent Information Systems*, 10:93–129, 1998. Cited on page 46.

[Rec10a] Jan Recker. Continued use of process modeling grammars: the impact of individual difference factors. *European Journal of Information Systems*, 19(1):76–92, 2010. Cited on pages 95 and 97.

[Rec10b] Jan Recker. Explaining usage of process modeling grammars: Comparing three theoretical models in the study of two grammars. *Information & Management*, 47(5):316–324, 2010. Cited on pages 95 and 98.

[Rec10c] Jan Recker. Opportunities and constraints: the current struggle with BPMN. *Business Process Management Journal*, 16(1):181–201, 2010. Cited on page 96.

[RL12] Jan Recker and Marcello La Rosa. Understanding user differences in open-source workflow management system usage intentions. *Information Systems*, 37:200–212, 2012. Cited on pages 94 and 95.

[RM05] Michael Rosemann and Michael zur Muehlen. Integrating risks in business process models. In *Proceedings of 5th Australasian Conferences on Information Systems (ACIS'05)*. AISeL, 2005. Cited on pages 28, 30, and 85.

[RMA07] Stefanie Rinderle-Ma and Wil M. P. van der Aalst. Life-cycle support for staff assignment rules in process-aware information systems. Technical Report WP 213, Eindhoven University of Technology, BETA Working Paper Series, 2007. Cited on page 134.

[RMBCO07] María Dolores R-Moreno, Daniel Borrajo, Amedeo Cesta, and Angelo Oddi. Integrating planning and scheduling in workflow domains. *Expert Systems with Applications*, 33(2):389–406, 2007. Cited on pages 4 and 46.

[RMTW11] Stefanie Rinderle-Ma, Farouk Toumani, and Karsten Wolf, editors. *Proceedings of the 9th International Conference on Business Process Management (BPM'11), Clermont-Ferrand, France, August 30 - September 2, 2011*, volume 6896 of *Lecture Notes in Computer Science*. Springer, 2011. Cited on pages 203, 213, and 226.

[RRD03] Manfred Reichert, Stefanie Rinderle, and Peter Dadam. On the common support of workflow type and instance changes under correctness constraints. In Robert Meersman, Zahir Tari, and Douglas C. Schmidt, editors, *Proceedings of On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE - OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003*, volume 2888 of *Lecture Notes in Computer Science*, pages 407–425. Springer, 2003. Cited on pages 44 and 46.

[RRD04] Stefanie Rinderle, Manfred Reichert, and Peter Dadam. On dealing with structural conflicts between process type and instance changes. In Desel et al. [DPW04], pages 274–289. Cited on pages 44 and 46.

[RSW13] Andreas Rogge-Solti and Mathias Weske. Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays. In Basu et al. [BPZF13], pages 389–403. Cited on page 52.

[RU00] R. Tyrrell Rockafellar and Stanislav Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–42, 2000. Cited on page 34.

[RWC+11] Kristian Rotaru, Carla Wilkin, Leonid Churilov, Dina Neiger, and Andrzej Ceglowski. Formalizing process-based risk with value-focused process engineering. *Information Systems and e-Business Management*, 9(4):447–474, 2011. Cited on pages 28 and 31.

[RWCN08] Kristian Rotaru, Carla Wilkin, Leonid Churilov, and Dina Neiger.

Formalising risk with value-focused process engineering. In Golden et al. [GAC+08], pages 1583–1595. Cited on pages 28 and 31.

[Sch00] Peter Schwartz. When good companies do bad things. *Strategy & Leadership*, 28(3):4–11, 2000. Cited on page 85.

[SEF+08] Kevin I. Smith, Richard M. Everson, Jonathan E. Fieldsend, Chris Murphy, and Rashmi Misra. Dominance-based multiobjective simulated annealing. *Evolutionary Computation, IEEE Transactions on*, 12(3):323–342, 2008. Cited on pages 64, 138, and 142.

[SGD05] Michael Schmitt, Bertrand Grégoire, and Eric Dubois. A risk based guide to business process design in inter-organizational business collaboration. In *International Workshop on Requirements Engineering for Business Need and IT Alignment (REBNITA 2005)*, 2005. Cited on pages 28 and 35.

[SGD+08] Prabhdeep Singh, Fatih Gelgi, Hasan Davulcu, Stephen S. Yau, and Supratik Mukhopadhyay. A risk reduction framework for dynamic workflows. In *Proceedings of the 2008 IEEE International Conference on Services Computing (SCC'08), Honolulu, Hawaii, USA, July 8-11, 2008*, pages 381–388. IEEE Computer Society, 2008. Cited on pages 6, 28, 35, and 53.

[SGF01] Gary Stoneburner, Alice Goguen, and Alexis Feringa. Risk Management Guide for Information Technology Systems Recommendations of the National Institute ofStandards and Technology. Technical report, USA/NIST, Oct 2001. Cited on pages 2 and 5.

[SHF10] Stefan Strecker, David Heise, and Ulrich Frank. RiskM: A multi-perspective modeling method for IT risk assessment. *Information Systems Frontiers*, pages 1–17, 2010. Cited on pages 28 and 32.

[Sie09] Amadou Sienou. *Proposition d'un cadre métholodogique pour le management intégré des risques et des processus d'entreprise.* PhD thesis, Centre de Génie Industriel, Mines Albi, Université de Toulouse, 2009. Cited on pages 28 and 29.

[Sim99] Robert L. Simons. How risky is your company? *Harvard Business Review*, 77(3):85, 1999. Cited on page 85.

[SKA12] Sjoerd van der Spoel, Maurice van Keulen, and Chintan Amrit. Process prediction in noisy data sets: A case study in a dutch hospital. In Philippe Cudré-Mauroux, Paolo Ceravolo, and Dragan Gasevic, editors, *Proceedings of the Second International Symposium on Data-Driven Process Discovery and Analysis (SIMPDA'12), Campione d'Italia, Italy, June 18-20, 2012*, volume 162 of *Lecture Notes in Business Information Processing*, pages 60–83. Springer, 2012. Cited on pages 132 and 134.

[SKLP08] Amadou Sienou, Achim P. Karduck, Elyes Lamine, and Hervé Pingaud. Business process and risk models enrichment: Considerations for business intelligence. In *IEEE International Conference on e-Business Engineering, 2008. ICEBE '08*, pages 732–735, oct 2008. Cited on pages 28 and 29.

[SKP06] Amadou Sienou, Achim P. Karduck, and Hervé Pingaud. Towards a framework for integrating risk and business process management. In Alexandre Dolgui, Gérard Morel, and Carlos E. Pereira, editors, *Information Control Problems in Manufacturing*, volume 6, pages 615–620. Elsevier, 2006. Cited on pages 28 and 29.

[SLKP07] Amadou Sienou, Elyes Lamine, Achim P. Karduck, and Hervé Pingaud. Conceptual model of risk: Towards a risk modelling language. In Mathias Weske, Mohand-Said Hacid, and Claude Godart, editors, *Proceedings of the Wise 2007 Workshops held in conjunction with the 8th International Conference on Web Information Systems Engineering (WISE'07), Nancy, France, December 3-6, 2007*, volume 4832 of *Lecture Notes in Computer Science*, pages 118–129. Springer, 2007. Cited on pages 28 and 29.

[SLP08] Amadou Sienou, Elyes Lamine, and Hervé Pingaud. A method for integrated management of process-risk. In Shazia Sadiq, Marta Indulska, Michael zur Muehlen, Xavier Franch, Ela Hunt, and Remi Coletta, editors, *Proceedings of the First Intenational Workshop on Governance, Risk and Compliance - Applications in Information Systems (GRCIS'08), held in conjunction with the 20th International Conference on Advanced Information Systems Engineering (CAiSE'08), Montpellier, France, June 17, 2008*, volume 339 of

CEUR Workshop Proceedings, pages 16–30. CEUR-WS.org, 2008. Cited on pages 28 and 29.

[SLPK08] Amadou Sienou, Elyes Lamine, Hervé Pingaud, and Achim P. Karduck. Towards a semi-formal modeling language supporting collaboration between risk and process manager. In *Proceedings of the Second IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST'08), Phitsanuloke, Thailand, February 26-29, 2008*, pages 119–125. IEEE, 2008. Cited on pages 28 and 29.

[SLPK09] Amadou Sienou, Elyes Lamine, Hervé Pingaud, and Achim Karduck. Aspects of the BPRIM Language for Risk Driven Process Engineering. In Robert Meersman, Pilar Herrero, and Tharam S. Dillon, editors, *Proceedings of On the Move to Meaningful Internet Systems: OTM 2009 Workshops, Confederated International Workshops and Posters, ADI, CAMS, EI2N, ISDE, IWSSA, MONET, OnToContent, ODIS, ORM, OTM Academy, SWWS, SEMELS, Beyond SAWSDL, and COMBEK 2009, Vilamoura, Portugal, November 1-6, 2009*, volume 5872 of *Lecture Notes in Computer Science*, pages 172–183. Springer, 2009. Cited on pages 28 and 29.

[SLPK10] Amadou Sienou, Elyes Lamine, Hervé Pingaud, and Achim P. Karduck. Risk driven process engineering in digital ecosystems: Modelling risk. In *4th IEEE International Conference on Digital Ecosystems and Technologies - Conference Proceedings of IEEE-DEST 2010, DEST 2010*, pages 647–650, 2010. Cited on pages 28 and 29.

[SM09] Alec Sharp and Patrick McDermott. *Workflow modeling: Tools for process improvement and applications development*. Artech House, 2009. Cited on page 47.

[Sma96] Clive Smallman. Risk and organizational behaviour: a research model. *Disaster Prevention and Management*, 5(2):12–26, 1996. Cited on page 85.

[SMBH11] Sebastian Schick, Holger Meyer, Markus Bandt, and Andreas Heuer. Enabling YAWL to Handle Dynamic Operating Room Management. In Florian Daniel, Kamel Barkaoui, and Schahram Dustdar, editors, *Proceedings Part II of the Business Process Manage-*

ment Workshops - BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, volume 100 of *Lecture Notes in Business Information Processing*, pages 249–260. Springer, 2011. Cited on pages 44, 45, and 46.

[SNP13] Camille Salinesi, Moira C. Norrie, and Oscar Pastor, editors. *Proceedings of the 25th International Conference on Advanced Information Systems Engineering (CAiSE'13), Valencia, Spain, June 17-21, 2013*, volume 7908 of *Lecture Notes in Computer Science*. Springer, 2013. Cited on pages 199 and 215.

[SOAH13] Suriadi Suriadi, Chun Ouyang, Wil M. P. van der Aalst, and Arthur H. M. ter Hofstede. Root cause analysis with enriched process logs. In La Rosa and Soffer [LS13], pages 174–186. Cited on pages 28 and 37.

[SSPC00] Apichart Suppapitnarm, Keith A. Seffen, Geoffrey T. Parks, and P. John Clarkson. A simulated annealing algorithm for multiobjective optimization. *Engineering Optimization*, 33(1):59–85, 2000. Cited on page 64.

[Sta04] Standards Australia and Standards New Zealand. *Standard AS/NZS ISO 4360*, 2004. Cited on pages 2, 5, 22, and 23.

[Sta09] Standards Australia and Standards New Zealand. *Standard AS/NZS ISO 31000*, 2009. Cited on pages 1, 2, 5, 6, 22, and 29.

[Sti09] Brett Stineman. IBM WebSphere ILOG Business Rule Management Systems: The Case for Architects and Developers. Technical report, IBM Software Group, Nov 2009. Cited on page 40.

[Sum03] Balram Suman. Simulated annealing-based multiobjective algorithms and their application for system reliability. *Engineering Optimization*, 35(4):391–416, 2003. Cited on page 64.

[Sum04] Balram Suman. Study of simulated annealing based algorithms for multiobjective optimization of a constrained problem. *Computers & Chemical Engineering*, 28(9):1849–1871, 2004. Cited on page 64.

[Swa98] Marianne Swanson. Guide for developing security plans for information technology systems. *NIST Special Publication*, 1998. Cited on page 2.

[SWDA08] M. Helen Schonenberg, Barbara Weber, Boudewijn F. Dongen, and Wil M. P. van der Aalst. Supporting flexible processes through recommendations based on history. In Dumas et al. [DRS08], pages 51–66. Cited on page 131.

[SWW+14] Suriadi Suriadi, Burkhard Weiß, Axel Winkelmann, Arthur H. M. ter Hofstede, Moe Wynn, Chun Ouyang, Michael J. Adams, Raffaele Conforti, Colin Fidge, Marcello La Rosa, and Anastasiia Pika. Current research in risk-aware business process management - overview, comparison, and gap analysis. *Communications of the Association for Information Systems*, 34, 2014. Cited on pages 2, 5, 6, 17, and 27.

[Syb11] Sybase. *Sybase CEP Implementation Methodology for Continuous Intelligence*, `http://www.sybase.com.au/files/White_Papers/Sybase_CEP_Implementation_Methodology_wp.pdf`. Accessed: Jun. 2011. Cited on pages 6, 42, and 101.

[TCG04] Kaijun Tan, Jason Crampton, and Carl A. Gunter. The consistency of task-based authorization constraints in workflow systems. In *Proceedings of the 17th IEEE Computer Security Foundations Workshop (CSFW'04), Pacific Grove, CA, USA, June 28-30, 2004*, page 155. IEEE Computer Society, 2004. Cited on page 159.

[TGM08] Paul Taylor, Jesus Jimenez Godino, and Basim Majeed. Use of fuzzy reasoning in the simulation of risk events in business processes. In LS Louca, Y Chrysanthou, Z Oplatkova, and K AlBegain, editors, *Proceedings of the 22nd European Conference on Modelling and Simulation (ECMS'08), Nicosia, Cyprus, 2008*, pages 25–30, 2008. Cited on pages 28 and 33.

[Tho10] Hedley Thomas. *Sick to Death*. Allen & Unwin, 2010. Cited on page 1.

[TJG+11] Simon Tjoa, Stefan Jakoubi, Gernot Goluch, Gerhard Kitzler, Sigrun Goluch, and Gerald Quirchmayr. A formal approach enabling risk-aware business process modeling and simulation. *Services Computing, IEEE Transactions on*, 4(2):153–166, 2011. Cited on pages 27 and 28.

[TJGK10] Simon Tjoa, Stefan Jakoubi, Sigrun Goluch, and Gerhard Kitzler. Planning dynamic activity and resource allocations using a risk-aware business process management approach. In *Proceedings of the Fifth International Conference on Availability, Reliability and Security (ARES'10), Krakow, Poland, February 15-18, 2010* [DBL10], pages 268–274. Cited on pages 27 and 28.

[TJGQ08] Simon Tjoa, Stefan Jakoubi, Gernot Goluch, and Gerald Quirchmayr. Extension of a methodology for risk-aware business process modeling and simulation enabling process-oriented incident handling support. In *Proceedings of the 22nd International Conference on Advanced Information Networking and Applications (AINA'08), GinoWan, Okinawa, Japan, March 25-28, 2008*, pages 48–55. IEEE Computer Society, 2008. Cited on pages 27 and 28.

[TJQ08] Simon Tjoa, Stefan Jakoubi, and Gerald Quirchmayr. Enhancing business impact analysis and risk assessment applying a risk-aware business process modeling and simulation methodology. In *Proceedings of the Third International Conference on Availability, Reliability and Security (ARES'08), Technical University of Catalonia, Barcelona, Spain, March 4-7, 2008*, pages 179–186. IEEE Computer Society, 2008. Cited on pages 9, 27, and 28.

[US 49] US Department of Defense. Procedures for performing a failure mode, effects and criticality analysis. November 1949. Cited on pages 24 and 30.

[UTO98] Ekunda L. Ulungu, Jacques Teghem, and C. Ost. Efficiency of interactive multi-objective simulated annealing through a case study. *Journal of the Operational Research Society*, 49:1044–1050, 1998. Cited on page 64.

[Ven07] David Vengerov. A reinforcement learning approach to dynamic resource allocation. *Engineering Applications of Artificial Intelligence*, 20(3):383–390, 2007. Cited on page 130.

[Vol11] Voluntary Interindustry Commerce Solutions Association. *Voluntary Inter-industry Commerce Standard (VICS)*. `http://www.vics.org`. Accessed: June 2011. Cited on pages 69 and 138.

[WA03] Antonius J. M. M. Weijters and Wil M. P. van der Aalst. Rediscovering workflow models from event-based data using little thumb. *Integrated Computer-Aided Engineering*, 10:151–162, Apr 2003. Cited on page 51.

[WA06] Janice Warner and Vijayalakshmi Atluri. Inter-instance authorization constraints for secure workflow management. In David F. Ferraiolo and Indrakshi Ray, editors, *Proceedings of the 11th ACM Symposium on Access Control Models and Technologies (SACMAT'06), Lake Tahoe, California, USA, June 7-9, 2006*, pages 190–199. ACM, 2006. Cited on page 159.

[WBL+11] Juliano Araujo Wickboldt, Luís Armando Bianchin, Roben Castagna Lunardi, Lisandro Zambenedetti Granville, Luciano Paschoal Gaspary, and Claudio Bartolini. A framework for risk assessment based on analysis of historical information of workflow execution in it systems. *Computer Networks*, 55:2954–2975, 2011. Cited on pages 28 and 36.

[Wer78] James R. Wertz, editor. *Spacecraft Attitude Determination and Control*. Kluwer Academic Publishers, 1978. Cited on pages 8 and 165.

[Wes07] Mathias Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer, 2007. Cited on page 19.

[Woo08] Gordon Woo. *Terrorism Risk. Wiley Handbook of Science and Technology for Homeland Security*, pages 2:1–2:17. John Wiley & Sons, 2008. Cited on page 1.

[WREW11] Di Wang, Elke A. Rundensteiner, Richard T. Ellison, and Han Wang. Active complex event processing infrastructure: Monitoring and reacting to event streams. In Serge Abiteboul, Klemens Böhm, Christoph Koch, and Kian-Lee Tan, editors, *Workshops Proceedings of the 27th International Conference on Data Engineering (ICDE'11), Hannover, Germany, April 11-16, 2011*, pages 249–254. IEEE, 2011. Cited on pages 42, 43, and 101.

[WRRM08] Barbara Weber, Manfred Reichert, and Stefanie Rinderle-Ma. Change patterns and change support features - enhancing flexibility

in process-aware information systems. *Data & Knowledge Engineering*, 66(3):438–466, Sept 2008. Cited on page 44.

[WW11] Burkhard Weiß and Axel Winkelmann. Developing a process-oriented notation for modeling operational risks - a conceptual metamodel approach to operational risk management in knowledge intensive business processes within the financial industry. In *Proceedings of the 44th Hawaii International International Conference on Systems Science (HICSS'11), Koloa, Kauai, HI, USA, January 4-7, 2011*, pages 1–10. IEEE Computer Society, 2011. Cited on pages 2, 28, and 30.

[WWB04] Barbara Weber, Werner Wild, and Ruth Breu. CBRFlow: Enabling Adaptive Workflow Management Through Conversational Case-Based Reasoning. In Peter Funk and Pedro A. González-Calero, editors, *Proceedings of the 7th European Conference on Advances in Case-Based Reasoning (ECCBR'04), Madrid, Spain, August 30 - September 2, 2004*, volume 3155 of *Lecture Notes in Computer Science*, pages 434–448. Springer, 2004. Cited on pages 44 and 160.

[WZM+11] Matthias Weidlich, Holger Ziekow, Jan Mendling, Oliver Günther, Mathias Weske, and Nirmit Desai. Event-based monitoring of process execution violations. In Rinderle-Ma et al. [RMTW11], pages 182–198. Cited on page 51.

[Yan08] I-Tung Yang. Utility-based decision support system for schedule optimization. *Decision Support Systems*, 44(3):595–605, 2008. Cited on pages 132 and 133.

[Yu97] Eric S. K. Yu. Towards modeling and reasoning support for early-phase requirements engineering. In *Proceedings of the Third IEEE International Symposium on Requirements Engineering (RE'97), Annapolis, MD, USA, January 5-8, 1997*, pages 226–235. IEEE Computer Society, 1997. Cited on page 34.

[ZBES07] Emmanuele Zambon, Damiano Bolzoni, Sandro Etalle, and Marco Salvato. A model supporting business continuity auditing & planning in information systems. In *Second International Conference on Internet Monitoring and Protection (ICIMP), San Jose, CA, USA*,

page 33, Los Alamitos, July 2007. IEEE Computer Society. Cited on page 30.

[ZD95] Wei Zhang and Thomas G. Dietterich. A reinforcement learning approach to job-shop scheduling. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95), Montréal Québec, Canada, August 20-25, 1995, 2 Volumes*, pages 1114–1120. Morgan Kaufmann Publishers Inc, 1995. Cited on page 130.