A Software Framework for Risk-Aware Business Process Management

Raffaele Conforti[1], Marcello La Rosa[1,2], Arthur H.M. ter Hofstede[1,4], Giancarlo Fortino[3], Massimiliano de Leoni[4], Wil M.P. van der Aalst[4,1], and Michael Adams[1]

[1] Queensland University of Technology, Australia
{raffaele.conforti,m.larosa,a.terhofstede,mj.adams}@qut.edu.au
[2] NICTA Queensland Lab, Australia
[3] Università della Calabria, Italy
g.fortino@unical.it
[4] Eindhoven University of Technology, The Netherlands
{m.d.leoni,w.m.p.v.d.aalst}@tue.nl

Abstract. With the large diffusion of Business Process Management (BPM) automation suites, the possibility of managing process-related risks arises. This paper introduces an innovative framework for process-related risk management and describes a working implementation realized by extending the YAWL system. The framework covers three aspects of risk management: risk monitoring, risk prevention, and risk mitigation. Risk monitoring functionality is provided using a sensor-based architecture, where sensors are defined at design time and used at run-time for monitoring purposes. Risk prevention functionality is provided in the form of suggestions about *what* should be executed, by *who*, and *how*, through the use of decision trees. Finally, risk mitigation functionality is provided as a sequence of remedial actions (e.g. reallocating, skipping, rolling back of a work item) that should be executed to restore the process to a normal situation.

1 Introduction

Companies are exposed to risks on a daily basis. Some of these risks may eventuate during the execution of business processes, for example due to a missed deadline or because the accrued cost may have exceeded an estimated budget threshold. We call this type of risk *process-related risk*. Ideally, every business process management system (BPMS) would provide process-related risk management capabilities covering the three main aspects of risk management: risk monitoring, risk prevention and risk mitigation. However, while commercial BPMSs [5,8] already provide monitoring functionalities that can be used for risk monitoring, nothing more than this is currently available. An extensive literature review in the area of risk-aware BPM has been carried out in [7], from it emerged how so far research on risk-aware BPM manly focuses on the design-time phase.

In this paper we present a framework[5] for risk management which not only provides features for risk monitoring but also for risk prevention and risk mitigation. The framework is composed of three main components, one for each aspect of risk management.

[5] available at http://www.yawlfoundation.org

The idea behind this framework is to enrich the business process management (BPM) life cycle with elements of risks management (see Fig. 1). Specifically, in the *Process Design* phase a business process model will be mapped with risks identified during the *Risk Identification* phase. During the *Process Implementation* phase these risks will be linked to specific elements of the process, e.g. resources (also known as actors), tasks, or data. In the *Process Enactment* phase, a risk-aware BPMS will execute the process, allowing the identification and treatment of risks during the *Process Diagnosis* phase.

Risk monitoring features are provided through a sensor-based architecture. Sensors are defined at design time by specifying a risk condition that needs to be monitored, and are activated at run-time by a sensor manager. The sensor manager also notifies the administrator when one of the sensors detects a risk. A risk eventuates if the risk likelihood associated with it exceeds a risk threshold defined at design time. The risk threshold indicates the risk level that a company is willing to accept.

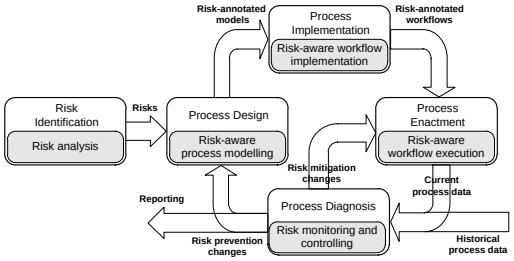

Fig. 1. Risk-Aware BPM Lifecycle

Risk prevention features are provided in the form of suggestions given to a resource while performing a process instance. The system, using decision trees, produces an estimation of the final risk level associated with each work item a resource may perform at a given moment, suggesting the work item with the lowest risk level to be executed. The system also provides support during the execution of a work item, by providing a predicted risk level based on the data values provided by a resource.

Finally, risk mitigation features are provided in the form of a mitigation strategy. When a notification is received, the administrator can request a mitigation strategy. In this case, the system uses a simulated annealing algorithm to identify a sequence of mitigation actions (e.g. reallocating, skipping, rolling back of a work item) that need to be executed in order to bring the process instance to a situation in which the risk likelihood is below the given threshold.

2 Risk Monitoring

Developed for flexibility, the sensor-based architecture provides for the monitoring of business processes via the definition of sensors. While the use of sensors to monitor business processes is not novel (see. e.g. [5]), the methodology behind the architecture is original, as is the functionality provided by the definition language [2].

The architecture allows monitoring through two phases (see Fig. 2): a design time phase, during which sensors are defined, and a run-time phase, during which a process instance is monitored by an instance of each sensor defined for such a process.

Conditions are defined through the use of variables and operations that specify what a sensor must monitor. These variables can be subdivided in three groups: i) *status*, providing information about status, times, number of execution, etc. of a work item;

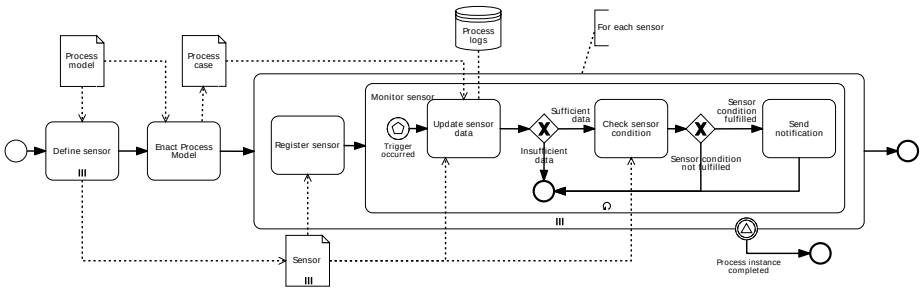

Fig. 2. Realization of risk-aware BPM lifecycle via sensors.

ii) *value*, the value of a data variable used during the execution of a process instance (for example the shipping cost of an order processed during the execution of an Issue Shipment Payment Order); and iii) *resource*, supplying information about the availability of a resource. Moreover, sensors are not limited to running instances, but can also collect information from completed instances and group them together to obtain aggregate information i.e. probabilities or frequencies. Finally, each sensor can have a trigger defined, which could be an event, e.g. work item completion, or a timer, e.g. every 5 seconds.

The runtime phase verifies and detects any changes to the process, while monitoring for the violation of specified conditions. Each condition is evaluated against the values the relevant variables have been assigned during the execution of the process.

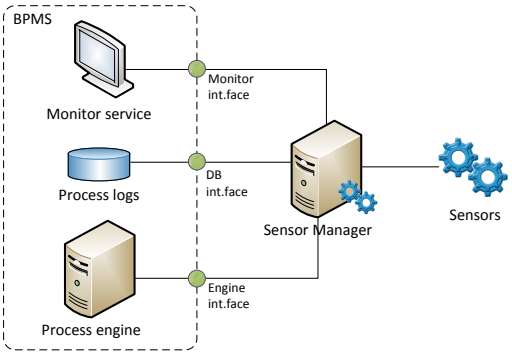

Fig. 3. System-Engine interactions

The architecture (see Fig. 3) is realized as a custom service of the YAWL Environment [9]. The YAWL Editor (see Fig. 4) was extended to provide new features that make it possible to easily identify variables and create risk conditions for the definition of new sensors, or to edit existing sensors. The editor also provides the ability to import risk conditions using a set, that can be extended by the user, of fourteen predefined templates. Figure 5 provides a screenshot of the wizard used during the creation of a risk sensor based on a template.

At the core of the architecture is the *Sensor Manager*, which provides for the creation of sensors and the interaction between sensors and YAWL system. The Sensor Manager contains three different interfaces, which interact with the YAWL engine, its logging database and the YAWL monitor service respectively. Once registered with the YAWL Engine, the Sensor Manager retrieves the process specification, initializes the sensors and keeps track of the execution of process instances. Periodically, using the update time specified at design time or whenever an event is triggered, the service notifies the sensors of the changes that have occurred in the process instance and the relevant data values

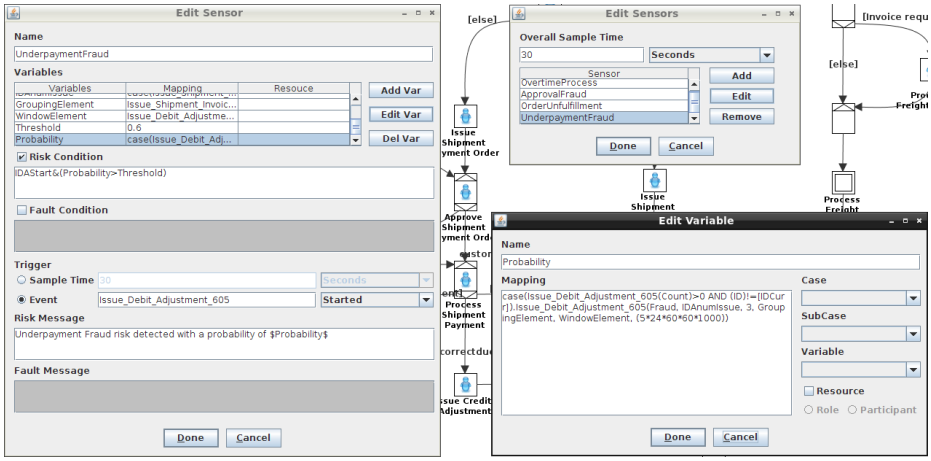

Fig. 4. The Risk Sensor Editor

retrieved from the process logs. Every time a sensor receives an update, it verifies the sensing condition. If the condition is violated the sensor notifies the Monitor Service which will notify the administrator marking the instance as “Risky”. Similarly, if the condition of an instance marked as “Risky” is no longer violated, the sensor will notify the Monitor Service which will mark the instance as “Safe”.

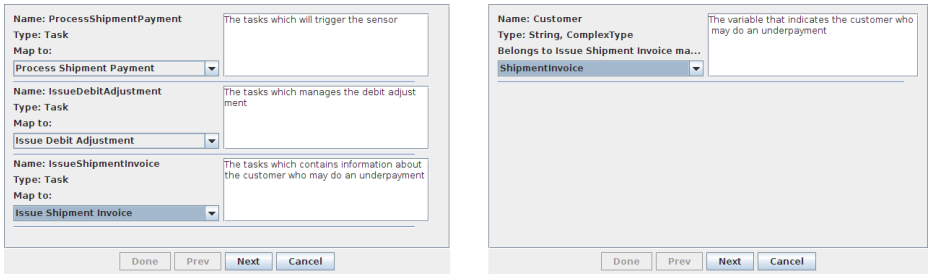

Fig. 5. The Wizard for the use of templates

3 Risk Prevention

Risk prediction functionality is provided in the form of support for risk-informed decisions during the execution of business processes. Each time a resource needs to execute (or is executing) a work item, the system provides information about the level of risk that will be reached at the completion of the process instance through the execution of that particular work item.

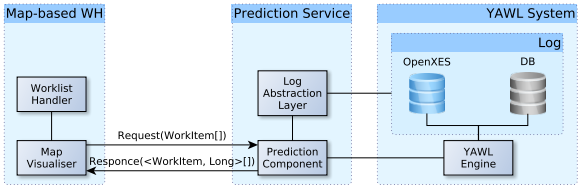

Fig. 6. The integration of the implemented tools with the YAWL system.

The system predicts the final level of risk by using decision trees as described in [1]. Once started, the system analyzes the log of the BPMS and generates a decision tree for each possible choice that may be taken (i.e. work item execution).

Figure 6 shows the architecture of this component. It is an independent service that on the one hand communicates with the BPMS, in this case the YAWL system, to retrieve the process model and its execution log, while on the other hand communicates with the resource handler to provide suggestions in the form or risk level associated with each work item. The process model is used to discover decision points (i.e. task), while the log is used to mine a decision tree for each decision point discovered, where each leaf of a decision tree represents a fault level that a process instance may reach at its completion.

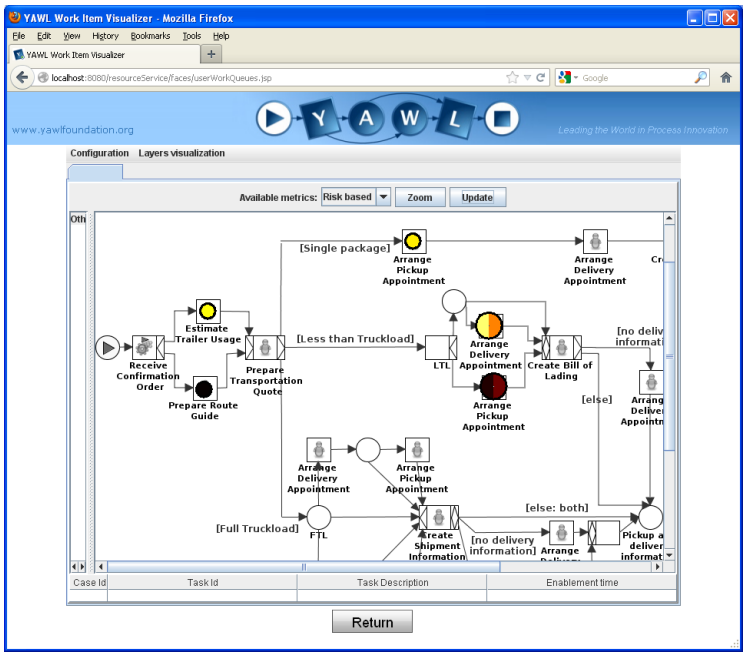

Fig. 7. A screenshot of the Map Visualizer

Once the component is initialized and all decision trees are built they are used at run-time to provide suggestions, which are visualized using a map-based workflow handler [4] (see Fig. 7). The suggestion visualizations are based on the expected risk level of each choice, based on the predicted fault level that will be reached by taking that choice times the frequency with which the prediction has previously proven correct.

During the execution of a process, a resource will be exposed to two types of suggestions. Figure 7 is a screenshot of the map-based workflow handler, where a resource will choose which work item to execute. The map-based workflow handler provides a graphical visualization of the process model, with dots on enabled work items. These dots represent the final risk level expected if the work item is executed. The color of each dot ranges from white, meaning a risk level equal to zero, to black, denoting a certain fault. These dots also grow in size, where the size of the dots reflects the number of work items actually enabled.

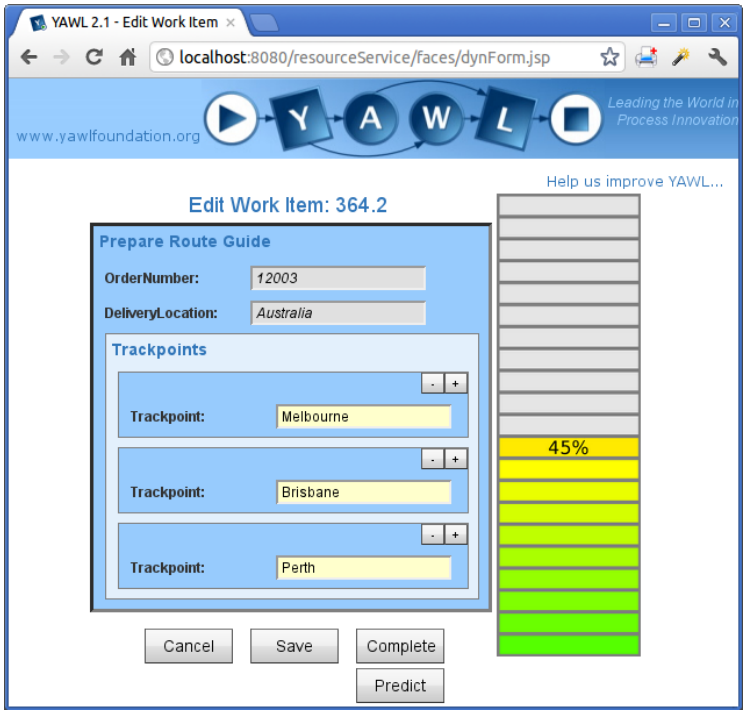

Fig. 8. A screenshot of the execution of a work item with risk prediction support

The second type of suggestion to which a resource can be exposed provides assistance during the execution of a work item, in particular with the completion of its form-based data fields. Figure 8 shows a screenshot taken during the completion of a form. By clicking on the "Predict" button each time a new value is entered into a form field, the system will update the graded bar on the right, indicating the revised final risk level.

To do so, the system will send the values entered on the form to the risk prevention component of the service, which will then use this information to navigate the decision tree associated with that particular work item and return the calculated risk level.

4 Risk Mitigation

The last component of our framework is the risk mitigation component, which interacts with the risk monitoring component and it is invoked by an administrator whenever a risk sensor detects a risk and the administrator considers it necessary to start a mitigation strategy.

Once a mitigation strategy is requested, this component will retrieve the status of each risk sensor involved in the monitoring of that specific process instance. These statuses are used to create a "virtual" environment in which mitigation strategies can be simulated in order to discover the optimal mitigation strategy. A mitigation strategy is composed of a sequence of mitigation actions that need to be executed in order to restore the process instance to a situation in which the risk no longer exceeds the tolerance threshold. These mitigation actions are atomic operations that work on the level of the single work item. Examples of mitigation actions are the rollback of a work item or the reallocation of a work item to another resource (for the complete list of mitigation action we refer to [3]).

To evaluate the best mitigation strategy, the risk mitigation component uses a simulated annealing algorithm [6]. The algorithm finds the sequence of mitigation actions that minimizes the risk level of each risk defined for that particular process, and the cost of the mitigation strategy (each mitigation action has a behavioral cost that needs to be considered). Finally, the best five solutions discovered are returned to the administrator who can then decide which one to apply.

5 Tool Maturity

The framework proposed in this paper has been tested with business process models from different domains, such as logistics, insurance and movie making processes, and with real and artificial logs. Each component of the framework exposes a different level of maturity. The monitoring component is the most advanced component. Tests have shown good results in terms of time performance, in general between a few milliseconds and a few seconds for the time required to identify a potential risk (based on the complexity of the risk condition) [2]. It was also used by several students in Italy and Australia in the context of university courses on process automation. This component was also tested with a sample of users in order to measure its usability and easy of use [2]. The results showed that the component helps modelers to understand the risks involved with a process, particularly when the modelers have limited experience. A limitation that arose from the study is that the language used for defining risk conditions was perceived as being too complex.

The risk prevention component was tested with artificial data and, at the time of writing, is being tested with real data. The utility of the visualization framework was tested in a separate work [4].

Finally, the risk mitigation component is the least mature one. It is currently not possible to automatically execute all the potential mitigation actions proposed as part of a mitigation strategy, and in particular a rollback on the system. This limitation means the component may propose a solution that cannot actually be executed. We are currently addressing this limitation.

6 Demo Script

This demo will provide an overview of the tool, showing each of the available features. The demo targets BPMS users such as business analysts and process participants, as well as researchers interested in the intersection between risk management and process management. The demo will show how to define a risk sensor from scratch using one of the available templates, and how to create a new template. After the definition of a risk sensor, a process instance will be executed to show how the system can support process participants in making risk-informed decisions by offering contextual suggestions, and how it is be able to detect potential risks and trigger warnings for process administrators. The demo will conclude with the discovery of a possible mitigation strategy.

Acknowledgments This research is funded by the ARC Discovery Project "Risk-aware Business Process Management" (DP110100091).

References

1. R. Conforti, M. de Leoni, M. La Rosa, and W.M.P. van der Aalst. Supporting risk-informed decisions during business process execution. In *Proc. of CAiSE*, LNCS. Springer, 2013. To appear.
2. R. Conforti, M. La Rosa, A. H. M. ter Hofstede, J. Recker, and M. Adams. Real-time risk monitoring in business processes: A sensor-based approach. BPM Center Report BPM-12-23, BPMcenter.org, 2012.
3. R. Conforti, A. H. M. ter Hofstede, M. La Rosa, and M. Adams. Automated risk mitigation in business processes. In *Proc. of CoopIS*, volume 7565 of *LNCS*. Springer, 2012.
4. M. de Leoni, M. Adams, W. M. P. van der Aalst, and A. 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.
5. Oracle. *BPEL Process Manager Developer's Guide*, http://download.oracle.com/docs/cd/E15523_01/integration.1111/e10224/bp_sensors.htm. Accessed: Jun. 2011.
6. K.I. Smith, R.M. Everson, J.E. Fieldsend, C. Murphy, and R. Misra. Dominance-based multiobjective simulated annealing. *Evolutionary Computation, IEEE Trans. on*, 12(3), 2008.
7. S. Suriadi, B. Weiß, A. Winkelmann, A. ter Hofstede, M. Wynn, C. Ouyang, M.J. Adams, R. Conforti, C. Fidge, M. La Rosa, and A. Pika. Current research in risk-aware business process management - overview, comparison, and gap analysis. BPM Center Report BPM-12-13, BPMcenter.org, 2012.
8. 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.
9. Arthur H. M. ter Hofstede, Wil M. P. van der Aalst, Michael Adams, and Nick Russell. *Modern Business Process Automation: YAWL and its Support Environment*. Springer, 2010.