Multi-perspective Business Process Monitoring

Amin Jalali and Paul Johannesson

Department of Computer and Systems Sciences, Stockholm University, Sweden
{aj,pajo}@dsv.su.se

Abstract. Monitoring business processes is an important area in Business Process Management. This area not only supports monitoring but also enables flexibility. Thus, it has been investigated in many other areas like Business Activity Monitoring, Exception Handling, Aspect Oriented Business Process Management, etc. These areas require to define how a process instance should be monitored from different perspectives. However, current definitions are coupled to control-flow perspective, which applies some limitations. For example, we cannot define a rule to capture situations in which an account balance is read - regardless of its process.

To capture such situations, we propose an approach to define monitoring rules. This approach enables composition of rules in a way to be decoupled from a specific perspective. To validate the result, we implemented a rule editor and a monitoring service, called Observer Service. These artefacts are used to support the definition of monitoring rules and track process instances, correspondingly. Finally, we investigated the validity and relevancy of the artefacts through a banking case study.

Keywords: Business Process Management Systems, Process Monitoring, Service Oriented Architecture, flexibility.

1 Introduction

Business Process Management(BPM) is an important area that supports automation of business processes. This automation is achieved through BPM life cycle including process design, configuration, enactment and diagnosis phases [28]. This life cycle resulted in rigid business processes that are not flexible. Therefore, another phase is added, called adjustment. In this phase, the enacted process can be adjusted and executed, without repeating the whole life cycle [3,16]. Both adjustment and diagnosis phases depend on monitoring process instances, that is achieved by tracking events. Each event is a possible monitoring point in BPM [23].

Two kinds of adjustment can be performed at runtime, i.e. allowing the process instance to be deviated from process specification, or changing the process specification and migrating the process instance(s) according to new specification. These adjustments are categorized under process flexibility as 'flexibility by deviation' and 'flexibility by change' [21]. These types depend on recognition of needed points (events) in process instances, which are fulfilled through process monitoring. Therefore, more capability in capturing events results in more ability in providing process adjustment and flexibility in action.

S. Nurcan et al. (Eds.): BPMDS 2013 and EMMSAD 2013, LNBIP 147, pp. 199–213, 2013.

The adjustment and flexibility should be performed for some monitoring points in a process instance. These points should be defined using some rules, called *monitoring rules*, that specify what information is a matter of interest for monitoring. Each event carries information related to different perspectives like control-flow (or process), task (or functional), operation (or application), data, resource (or organizational), time, etc [1,20].

There is an implicit order between perspectives when defining a process model. J. Becker et al, mention that "[t]he data flow is restricted by the control flow as the data flow cannot precede the control flow" [10]. These restrictions are valid in process definition. However, they limit process monitoring if we want to define the monitoring rules with the same approach.

For example, a bank manager might be interested to monitor high value financial transactions. These transactions can be occurred by different processes and activities. If we want to define monitoring rules using the control-flow and task perspectives, we should define a lot of monitoring rules to capture each task to investigate if the value of the transaction is more than the limit.

In this paper, we solve this problem by proposing an approach to define monitoring rules independent of any specific process perspective. To do that, we introduce possible monitoring points in a process instance. Then, we investigate what sort of information can be found for each point and the relation between them. As a result, we define an algorithm to evaluate monitoring rules from different perspectives. To validate our result, we developed two applications, i.e. the monitoring rule editor and the Observer Service. The editor supports the definition of monitoring rules based on investigated relation. The service monitors process instances to check if rules are satisfied or not. Moreover, we investigated the relevancy of the problem using a banking case study. The actual implementation of the study using our artifices is in progress.

Therefore, the remainder of the paper is organized as follows. In Section 2, we give a description of how the rules are defined in different BPM areas. Then, we present the relation between process perspectives and different states of monitoring life cycles in Section 3. Section 4 demonstrates the rule editor and the architecture of the implemented service. Section 5 investigates the validity and relevancy of the artefacts using a banking case study. Section 6 discusses the limitations of the work. Finally, Section 7 presents directions for future works and concludes the paper.

2 Background

A lot of areas in BPM paradigm try to define how monitoring points should be specified such as *Business Activity Monitoring(BAM)*, *Exception Handling*, *Aspect Oriented Business Process Management(AOBPM)*, etc. Therefore, different attempts have been performed to monitor these points. In this section, we look at some of these attempts in general.

Business Activity Monitoring(BAM) is defined by Gartner as a concept which enables tracking business operations and making issues visible quickly, based on

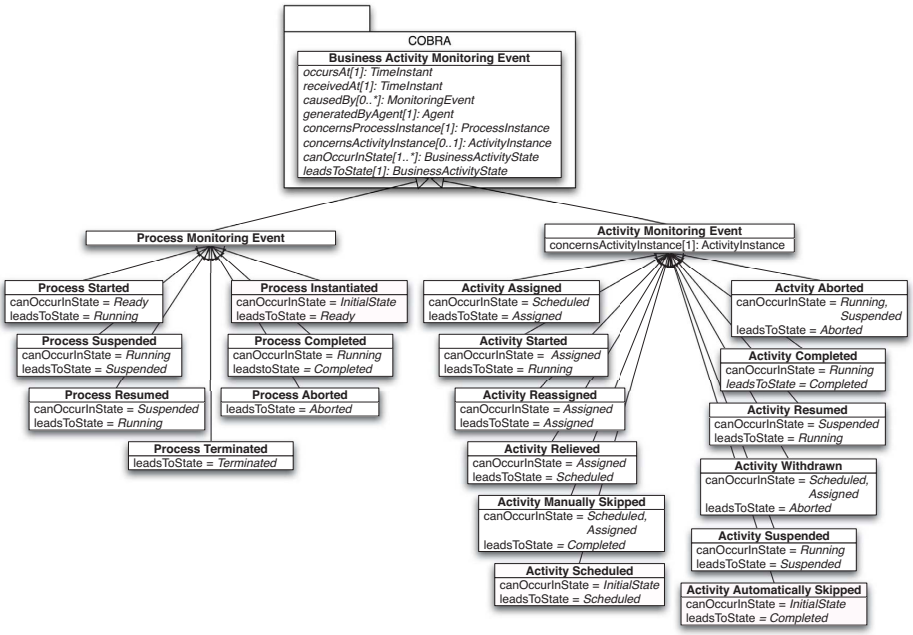

Fig. 1. Events Ontology [22]

real-time data[1]. BAM could be implemented through different approaches such as process mining [27] and real-time monitoring [19]. Process mining enables tracking of business operations using event logs [7]; while, real-time monitoring detects different monitoring points in process instances and keeps track of them in action. The process mining is out of our attention, because we focus on how to define rules to capture various monitoring points in this paper. Definition of rules in BAM has been investigated in different research, e.g. [13,15,26]. For example, Pedrinaci et. al. define an event ontology for BAM in [22] (see Figure 1). The events are categorized into two groups for monitoring, i.e. process and activity. The process monitoring event consists of started, suspended, resumed, terminated, aborted, completed and instantiated events. The activity monitoring event includes assigned, started, reassigned, relieved, skipped, scheduled, aborted, completed, resumed, withdrawn, suspended and skipped. These events are points that a process instance can be monitored. However, it does not mean that we have to restrict the definition of rules to them. Such restriction can end up us with current limitations in defining monitoring rules, which are coupled to control-flow and task perspectives.

Exception Handling is an important area of BPM, which tries to support flexibility by deviation [2]. Deviations are recognized by tracking some monitoring points and evaluating some exception rules. Monitoring points are categorized into five groups, i.e. Work Item Failure, Deadline Expiry, Resource Unavailability,

[1] `http://www.gartner.com/it-glossary/bam-business-activity-monitoring`

External Trigger and Constraint Violation [8,24]. Each points can be looked from different perspectives. For example, the task that the work item is going to perform represents the functional perspective. The resource to whom the work item is allocated represents the resource perspective. Exception rules specify when an exception should be captured. Again, definition of exception rules are coupled with control-flow perspective. For example, these rules are defined using an exception event attached to an activity's boundary in Business Process Model and Notation (BPMN) [14].

Moreover, the same approach can be tracked for handling exceptions in other works like worklet [9], where the exception types could be recognized in a process instance using a set of rules, called Ripple Down Rules(RDR) [8]. These rules could be defined to monitor violations of exception types, which happens for a specific case or a workitem. This means that the constraint should be mapped to control-flow or task perspective first. This approach limits the definition of rules, which needs to be independent of these two perspectives. It means that, if we want to define a monitoring point for a resource or data perspective, we have to map it to control-flow or task perspectives. For example, if a customer wants to get notified when his or her account balance is decreased, we should find all cases and tasks which have the potential to withdraw money from the customer account. This limitation is because we are not able to define a constraint for only data perspective here.

Aspect Oriented Business Process Management(AOBPM) aims to separate cross-cutting concerns from business processes and model them separately [11,12,17]. Separated models need to be weaved at runtime. The weaving requires to check monitoring points to see whether some aspects are specified for them [18]. Monitoring points are called *join points*, and rules are called *pointcuts*. The rules are defined based on control-flow and task perspectives. This implies some limitations in defining aspects, e.g. enforcing some security policy when an extra confirmation is required for activities of a new clerk. Again, we need to define rules for all tasks of a process instance to check if a new clerk performed it or not.

To sum up, we found a general approach in definition of monitoring rules in different areas of BPM. These rules are defined based on control-flow and task perspectives. Other perspectives can be incorporated in rules when one of these two perspectives are specified. This implies a lot of limitations for defining business monitoring points for process instances. In fact, it enforces multiple definition of rules to capture a business monitoring point. Therefore, we define a new approach to capture monitoring points and defining rules. In the next section, we introduce this approach.

3 Approach

Monitoring points can be defined with the help of workitem life cycle. The life cycle is defined by N. Russell et al [25], and it is general for different Workflow Management Systems(WfMSs) [29]. It consists of the states that a workitem can have in its life. States can be changed by transitions(events) (look at the dashed

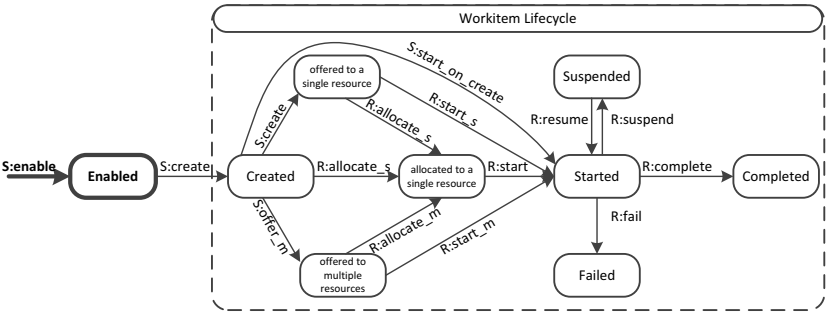

Fig. 2. Workitem Monitoring life cycle

rectangle in Figure 2). Transitions' names are started with R or S, representing whether a resource or the system(WfMS) initiates the transition. A. Rogge-Solti et al, define two more states for a workitem life cycle, i.e. *init* and *enabled* [23]- although they do not consider some other states. We could not consider these states as workitem life cycle states, since the workitem is not created yet. We also could not find any application of init state, so we exclude it. However, we consider the enabled state as one of the wokitem monitoring states because it is important from the monitoring viewpoint. Therefore, we end up with a new life cycle for monitoring, which is shown in Figure 2

The life cycle starts when the WfMS detects a workitem as enabled. All enabled workitem will not be created. For example, when a process model contains a deferred choice, many workitems can be enabled. However, as soon as one of them is created, others will not be enabled any more [5]. The created workitem can be 'offered to a single resource', 'offered to multiple resources' or 'allocated to a single resource'. Moreover, it could be started by the WfMS if it is an automated workitem. The started workitem can be suspended, and the suspended workitem can be resumed. The started workitem can also be failed or completed.

Furthermore, we should monitor the process instances, called cases. We recognized different states, which can be monitored during a case life cycle, i.e. created, completed, suspended and failed. These states could only be changed by the WfMS, so we do not incorporate the name of event initiators in event labels.

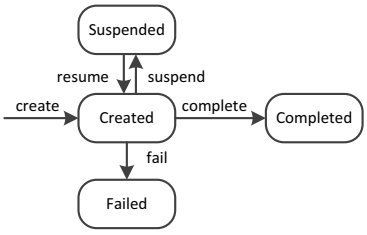

Fig. 3. Case Monitoring life cycle

	Perspectives				
Status	Control-flow	Data		Task	Resource
Case Monitoring Lifecycle		Case level	Workitem level		
Created	+	+ (r)	-	-	-
Suspended	+	+	-	-	-
Failed	+	+	-	-	-
Completed	+	+ (w)	-	-	-
Workitem Monitoring Lifecycle					
Enabled	+	+	-	+	-
Created	+	+	+ (r)	+	-
offered to a single resource	+	+	+	+	+
offered to multiple resources	+	+	+	+	+
allocated to a single resource	+	+	+	+	+
Started	+	+	+	+	+
Suspended	+	+	+	+	+
Failed	+	+	+	+	+
Completed	+	+	+ (w)	+	+

Fig. 4. The relation between level, states and perspectives

Although there is other states during execution of a case and workitem like cancelled, there are different from one WfMS to the other. Thus, we consider the general states which exist in most of WfMSs and limit our Monitoring life cycles to existing states (see Figure 2 and Figure 3). The Workitem and Case Monitoring life cycles can be used to define monitoring points from different business process perspectives. Each process model consists of different perspectives, and each perspective exposes a different kind of monitoring points , so we should investigate what sort of monitoring points exist in each workitem monitoring state to define monitoring rules.

Figure 4 shows what monitoring points could be tracked in each state of Case and Workitem Monitoring life cycle. The control-flow monitoring points can always be captured in both life cycles. The data perspective is restricted by control-flow perspective in a way that it is always defined based on control-flow perspective [10]. Thus, we divide data perspective into two sub-categories, i.e. case level and workitem level. The case level data can be accessed in all states of both life cycles. The reading operation of data is performed when the case is created, so we added '(r)' to demonstrate this fact in Figure 4. The writing operation of data is performed when the case is completed, which is shown by '(w)' in the figure. The workitem level data can be accessed in all states of workitem life cycles, but it cannot be accessed in the enabled state of workitem monitoring life cycle.

Furthermore, The task monitoring points are available during workitem monitoring life cycle. The resource monitoring points are also available during all states of workitem monitoring life cycle except enabled and created. In these states, the resource is not yet offered or allocated, so there is no information about who will carry on the workitem.

Figure 4 indicates how the rules can be defined for monitoring process instances from different process perspective. For example, if we want to define a rule to enforce confirmation of all works that have been done by a new clerk, we should define it as intersect of workitem completed state and resource

perspective in the figure. Other cells in the same row indicate that we can limit this rule based on other perspectives. For example, we can limit this rule to enforce confirmation if the amount is greater than a limit. It is possible because we have the data perspective in this row.

To evaluate these rules, we define an algorithm which shows how different monitoring points can be analyzed. This algorithm uses basic terms, which are given as follow. The terms start with definition of perspectives, which can have any number of members. Thus, new perspectives can easily be added as a new member without changing the algorithm.

Definition 1 (Basic Definition).

- $\mathcal{P} = \{\mathit{Control\text{-}flow}, \mathit{Data}, \mathit{Task}, \mathit{Resource}\}$ *is a set of Perspectives. Here, we limit ourself to four perspectives, but they can be added just as a member of the set.*
- $\mathcal{L} = \{\mathit{Case}, \mathit{Workitem}\}$ *is a set of Levels. There are two levels for monitoring, i.e. Case and Workitem. Case represents the executed instance of a process model. The workitem is an executed instance of a task.*
- $\mathcal{W}_{en} = \{\mathit{s{:}enable}, \mathit{s{:}create}, \mathit{s{:}start\_on\_create}, \mathit{s{:}offer\_m}, \mathit{R{:}start\_s}, \mathit{R{:}allocate\_s}, \mathit{R{:}allocate\_m}, \mathit{R{:}start\_m}, \mathit{R{:}suspend}, \mathit{R{:}resume}, \mathit{R{:}fail}, \mathit{R{:}complete}, \}$ *is a set of WorkItem Event Names. These names are derived from Workitem life cycle. We also added s:enable to monitor the workitem when it gets enabled.*
- $\mathcal{C}_{en} = \{\mathit{create}, \mathit{suspend}, \mathit{resume}, \mathit{complete}, \mathit{fail}\}$ *is a set of Case Event Names.*
- $\mathcal{E}_n = \mathcal{C}_{en} \bigcup \mathcal{W}_{en}$ *is a set of Event Names, which is a union of case event names and workitem event names.*
- $\mathcal{E}_d = (\mathcal{P}, Value)$ *is EvenetData, which is a tuple. It contains a perspective and its values.*
  – Value is a simple string. This string can contain, for example, xml representing the data perspective.
- $\mathcal{E}_{ds} = \{\mathcal{E}_d\}$ *is a set of Event Data.*
- $\mathcal{C} = \mathcal{E}_d$ *is Condition. The condition is an EventData, which is a tuple containing a perspective and its values. We distinguish between event data and condition because event data is what happened in execution; while, condition is abstract representation of the situation that should be monitored.*
- $\mathcal{C}_s = \{\mathcal{C}\}$ *is a set of Condition.*
- $\mathcal{E} = (\mathcal{E}_n, \mathcal{E}_{ds})$ *is Event. The event is a tuple. It includes an event name and a set of event data. In this way, each event can carry different data from different perspectives.*

Definition 2 (Monitoring Rules Definition).

- $\mathcal{M} = (\mathcal{L}, \mathcal{E}_n)$ *is MonitoringPoint. It is a tuple, which contains a level and an event name. It means that a monitoring point can be any event in case or workitem level.*
- $\mathcal{M}_s = \{\mathcal{M}\}$ *is a set of Monitoring Points.*
- $\mathcal{R} = (\mathcal{M}, \mathcal{C}_s)$ *is Rule, which is a tuple. It contains a monitoring point and a set of conditions. It means that a rule defines the criteria that capture monitoring points, which can be limited from different perspectives.*
- $\mathcal{R}_s = \{\mathcal{R}\}$ *is a set of Rules (Ruleset).*

The ruleset is used by the observer service to determine if an event satisfies conditions or not. 'Algorithm 1' shows how events can be examined to see if conditions in the ruleset are satisfied. The algorithm gets the event, level and the ruleset. It checks the rules based on specified conditions in the ruleset, and returns the set of rules, which are satisfied. The condition can have '*' as the value, which means that all values can be accepted. This algorithm is not designed for any specific perspective, so it is general. As a result, by adding any perspective to the set of perspectives the algorithm will not be changed. This algorithm are implemented in the Observer service which is described in the next Section.

Algorithm 1. Evaluate Monitoring Rules

Input: l:$\mathcal{L}$,e:$\mathcal{E}$,rs:$\mathcal{R}_s$
Output: $\mathcal{R}_s$

```
 1: Rs result;
 2: for each R r in rs do
 3:    if r.M=(l,e) then
 4:       Boolean ruleResult := true;
 5:       for each C c in r.Cs do
 6:          if ruleResult=true AND c.value<>'*' then
 7:             for each Ed ed in e.Eds do
 8:                if ed.P=c.P AND ed.Value<>c.Value then
 9:                   ruleResult := false;
10:                end if
11:             end for
12:          end if
13:       end for
14:       if ruleResult=true then
15:          result.Add(r);
16:       end if
17:    end if
18: end for
19: return result;
```

4 Implementation

To enable definition of rules in a way that supports all combinations, we developed a rule editor and an Observer service.[2] In this section, we describe the rule editor and the architecture of the service.

4.1 Rule Editor

The rule editor is designed in a very generic way that can be extended easily to support other states and perspectives. It reads the perspectives and states

[2] Both the rule editor and the Observer Service can be downloaded from `http://people.dsv.su.se/~aj/ObserverService/`

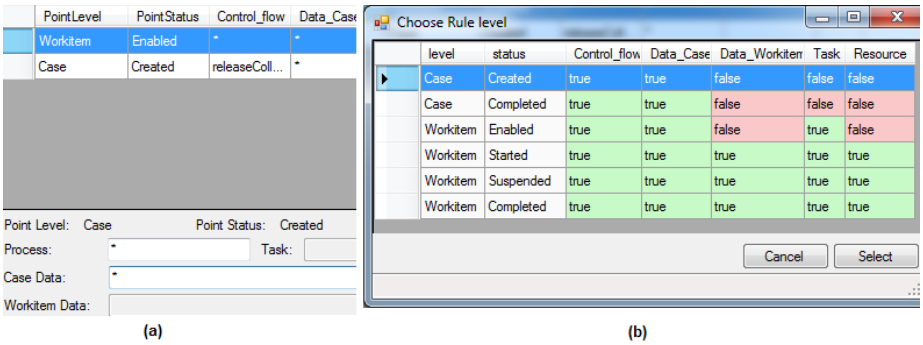

Fig. 5. RuleEditor

from an xml file. The xml indicates what information exists in each event. The result is shown to the user when s(he) wants to define a monitoring rule (see Figure 5(b)). This window shows a table which is similar to Figure 4. It consists of possible levels and states for monitoring points. For each state, the editor shows what perspectives are available to be limited by monitoring rules. For example, Data_Workitem and Resource are not available for workitem enabled monitoring points. This awareness supports users to define rules, which comply to the context of events.

The user can limit the data for each perspective in the editor(see Figure 5(a)). To do that, the user should select the level (workitem or case) and the state for which s(he) wants to observe the process. The editor enables the user to apply some limitation for the monitoring point based on information that exists on that point. This information can be limited from different perspectives.

For example, the bank manager might be interested to monitor all tasks that have been done by a specific clerk if s(he) works on collateral which worth more than 1,000,000 USD in all processes. S(he) should select the row from the table that has 'Workitem' as level and 'Completed' as the state. Then, the editor recognizes what information is available in that state, i.e. Control-flow, Data (both in case and workitem levels) and Resource. The control-flow should not be limited to any process, so '*' should be written - which indicates all processes. The Case-data should not also be limited, so '*' should be written. The Workitem-data should be limited to 1,000,000, so an xpath can be written to check the data condition, i.e. '//Collateral/Amount >1000000'. The Resource should also be limited to the specific clerk, so the name of the clerk can be written in Resource section.

This editor writes all rules in an XML file, that is used by Observer Service to monitor process instances. We limited the user to select the level and states when defining the rule. However, if the user is interested to define a rule for all levels or states, (s)he can still do that by changing the level or state field to '*' in the XML file. The architecture of the service is explained in the next section.

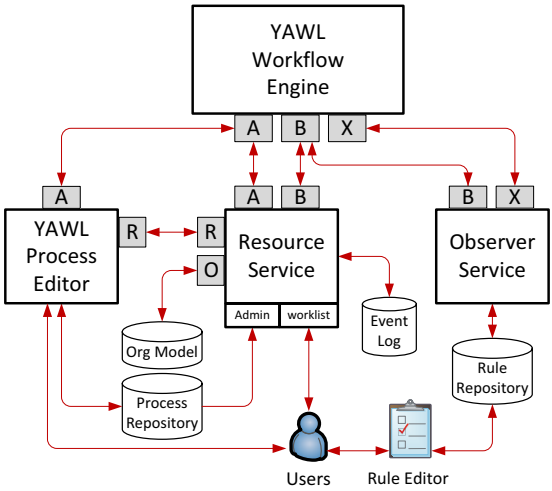

Fig. 6. Observer Services Architecture

4.2 Architecture

The Observer Service is responsible to track process instances based on monitoring rules, which are composed using the editor. The service is designed based on Service Oriented Architecture. It monitors process instances using events, which are received from WfMS. Therefore, it can be configured to observe any WfMS. We chose YAWL as a WfMS for which we monitor process instances. YAWL is selected since it supports full workitem life cycle, and it supports many workflow patterns. It also has formal definition and semantic. Moreover, it is open-source and is developed based on Service Oriented Architecture [4,6].

Figure 6 shows the architecture of our service and its relation to the WfMS. The service is connected to the YAWL Engine through two interfaces, i.e. B and X. Interface X is used to monitor case monitoring points and 's:enable' event from workitem monitoring life cycle; while, interface B is used to monitor workitem life cycle.

The resource service also plays an important role here. It is responsible to offer and allocate workitems to users. Therefore, it initiates changing some workitem states. This service collaborates with the YAWL engine through interface B to change the state of Workitems. The Interface A is utilized to upload specification to the engine, when a user launches a new process. The resource service also reads the organizational model through Interface O, which can be used for extending monitoring rules.

The Observer Service does not track other services; instead, it tracks the changes in workitem and case states through the engine. The service also reads the rules (composed by the Rule Editor) from Rule Repository. The rules specify which events should be captured. In the next section, we describe the case study which we conducted to investigate the relevancy of the artefact.

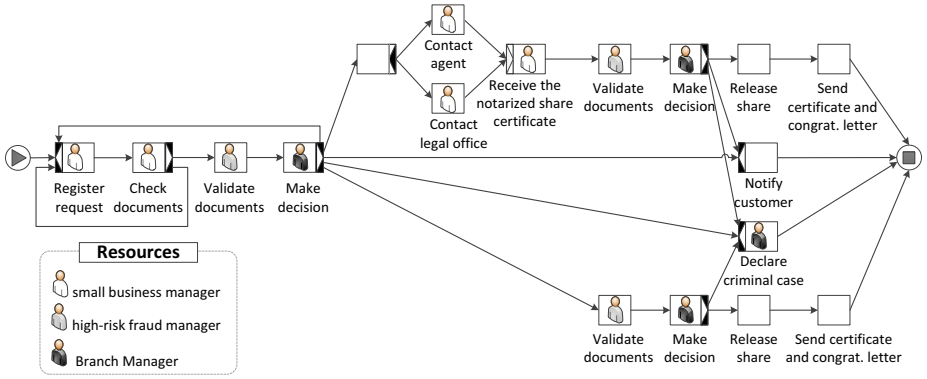

Fig. 7. Release Collateral

5 Case Study

In this section, we validate the relevance of our artefact using a banking case study. In this study, we considered different banking process, among them we chose the release collateral business process. The aim of the process is to release the collateral when the debt is not fully paid. The bank can decide about releasing the collateral based on the customer record. There are many types of collateral such as share, stock, warrant, option and co-signer. We excluded co-signer collateral since it makes the process much more complex. This complexity makes presentation of the process not possible in this paper.

Figure 7 shows this process model from the control-flow perspective. The process starts when a small business manager receives and registers documents from customer to release his or her collateral. Then (s)he checks all supporting documents. If a document is missed, then (s)he asks the customer to send complementary documents. When all documents are collected, they are sent to the high-risk fraud manager for additional review and validating the originality of documents. It is a security policy in the bank to check the originality of all documents which are received from other parties, except other banks. Then the branch manager comes up with any of the following situations:

- Declaration of criminal case: If the high-risk fraud manager detects a document as fake, the branch manager will declare the case as criminal.
- Completion of documents: The branch manager may ask for more supporting documents before any decision made to the collateral release.
- Rejection: If s(he) decides the rejection, then the customer will be notified.
- Acceptance: if s(he) accepts the release request, then two different activities can be performed depending on the type of the collateral:
  - If the collateral is stock, warrant or option, then the small business manager contacts the investment brokerage office and receives required information such as the most current investment activity statement. Then, the branch

manager decides on the case, i.e. rejection or acceptance. If s(he) rejects the case, the customer will get notified; otherwise, the requested collateral will be released. Then, the congratulation letter will be sent to the customer.
– If the collateral is a share, then the small business manager contacts the Investment broker and the lawyer, who had originally published the share certificate to the customer. When s(he) receives the notarized share certificate, the high-risk fraud manager should validate the originality of the document, due to security policy. If the high-risk fraud manager detects the document as fake, then the branch manager will declare the case as criminal. Otherwise, the business manager accepts the case, in most of the cases. However, if s(he) rejects the case, the customer will get notified. In case of acceptance, the share collateral will be released, and the certificate and congratulation letter will be sent to the customer.

The business manager might be interested to get notified if someone works on collateral, which has very high value. The high value is subjective and can be varied in times, so it should be determined by the business manager. Currently, we have to define monitoring points for all activities to capture such events. However, with our artefact we can get this kind of alarm by defining one rule.

The rule is defined as:

```
<rule process='releaseAsset' task='*' state='wi.completed'
data='\\Collateral \Amount >1000000' />
```

The modelling phase of this case study is finished, and the implementation phase is still on progress. We also found out that if tasks can be categorized, it would be highly beneficial for defining monitoring points. For example, a bank manager might be interested to monitor all payment tasks, or all financial tasks. If tasks' types can be defined in process models, it can be also used when tracking them.

6 Limitations

We applied different limitations in different steps of this research such as in solution, implementation and case study.

In solution, we limit ourselves to general workitem life cycle. This means that we did not consider some states of case and workitem which are not general in WfMSs. For example, we did not consider cancelled state in both of life cycles. Moreover, we did consider limited number of perspectives to define our solution, i.e. Control-flow, Data and Resource. Therefore, our solution does not cover other perspectives like time. Although these limitations restrict our solution, it does not affect the research outcome. It is due to the fact that the solution is general and can be easily expanded to support other states and perspectives.

In implementation, we did not consider resource perspective because the YAWL engine is not responsible for that. Indeed, the resource service handles this responsibility, and the YAWL engine cannot track changes of states, which are performed by other services. If such state are going to be considered, the resource service of YAWL engine should be tracked, instead of the YAWL engine.

As a result, we had to dismiss 'offered to a single resource', 'offered to multiple resource', 'allocated to a single resource' states. This limitation does not affect the result, because other perspectives are implemented and investigated. Moreover, the implementation is general and can be easily extended to support other perspectives if the WfMS supports it.

In case study, we exclude one sort of collateral, which is used in the bank, due to reducing complexity for presentation. This limitation does not affect our goal, since we wanted to show the relevancy of the problem in the real domain. Moreover, we currently finished the modelling part of the case study, and it is not implemented completely. The implementation is in progress.

7 Conclusion and Future Works

In this paper, we presented a generic solution to monitor process instances from different business process perspectives. This solution recognizes the events as possible points that can be tracked. Each event carries different information from a different perspective. Therefore, we consider what sort of information from what perspective can be monitored in what event. The result was a set of relations that shows how the rules can be defined to comply with the process content. To validate the result, we developed two applications, i.e. a rule editor and an Observer Service. The rule editor supports definition of rules based on the defined relations. The Observer Service monitors business processes based on rules, which are defined by rule editor. In this way, we can capture events, which might be interested from a different perspective.

The relevancy of artefact is investigated by a case study in a banking domain. In this study, we chose 'releasing collateral' process. This process is used to release collateral when the customer has not paid his or her dept completely. The monitoring rule can restrict monitoring points to those in which someone works on a collateral having a high value. Despite other approaches that need to define a lot of rules for each activity to capture this business event, our artefact monitors it just by one rule. Moreover, we also distinguished the following future works:

- providing features that enable other services to subscribe for a special event.
- Using Business Rule Management System (BRMS) to define and analyse more complex monitoring rules.
- Adding more perspectives when defining rules, e.g. time, resource, cost, etc.
- Considering how our artefact can handle more exception handling in process models.
- Enabling Aspect Oriented Business Process Execution based on this artefact and investigating how much it supports separation of concerns from different perspectives.
- Considering how our artefact can extend the Business Activity Monitoring in terms of defining more measures.

Acknowledgement. We would like to appreciate Mrs. Marjan Taheri from Actax Inc. for her valuable helps in our case study. We also thank Dr. Petia Wohed to give us valuable feedback on this work.

References

1. van der Aalst, W.M.P.: Workflow verification: Finding control-flow errors using petri-net-based techniques. In: van der Aalst, W.M.P., Desel, J., Oberweis, A. (eds.) Business Process Management. LNCS, vol. 1806, pp. 161–183. Springer, Heidelberg (2000)
2. van der Aalst, W.M.P., Adams, M., ter Hofstede, A.H.M., Pesic, M., Schonenberg, H.: Flexibility as a service. In: Chen, L., Liu, C., Liu, Q., Deng, K. (eds.) DASFAA 2009 Workshops. LNCS, vol. 5667, pp. 319–333. Springer, Heidelberg (2009)
3. van der Aalst, W.M.P., et al.: Process mining manifesto. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM 2011 Workshops, Part I. LNBIP, vol. 99, pp. 169–194. Springer, Heidelberg (2012)
4. van der Aalst, W.M.P., Aldred, L., Dumas, M., ter Hofstede, A.H.M.: Design and implementation of the YAWL system. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 142–159. Springer, Heidelberg (2004)
5. van der Aalst, W.M.P., Barros, A.P., ter Hofstede, A.H.M., Kiepuszewski, B.: Advanced workflow patterns. In: Scheuermann, P., Etzion, O. (eds.) CoopIS 2000. LNCS, vol. 1901, pp. 18–29. Springer, Heidelberg (2000)
6. van der Aalst, W.M.P., ter Hofstede, A.H.M.: Yawl: yet another workflow language. Information Systems 30(4), 245–275 (2005)
7. van der Aalst, W.M.P., Weijters, A.: Process mining: a research agenda. Computers in Industry 53(3), 231–244 (2004)
8. Adams, M., ter Hofstede, A.H.M., van der Aalst, W.M.P., Edmond, D.: Dynamic, extensible and context-aware exception handling for workflows. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 95–112. Springer, Heidelberg (2007)
9. Adams, M., ter Hofstede, A.H.M., Edmond, D., van der Aalst, W.M.P.: Worklets: A service-oriented implementation of dynamic flexibility in workflows. In: Meersman, R., Tari, Z. (eds.) OTM 2006. LNCS, vol. 4275, pp. 291–308. Springer, Heidelberg (2006)
10. Becker, J., Rosemann, M., von Uthmann, C.: Guidelines of business process modeling. In: van der Aalst, W.M.P., Desel, J., Oberweis, A. (eds.) Business Process Management. LNCS, vol. 1806, pp. 30–49. Springer, Heidelberg (2000)
11. Cappelli, C., Santoro, F.M., do Prado Leite, J.C.S., Batista, T., Medeiros, A.L., Romeiro, C.S.C.: Reflections on the modularity of business process models: The case for introducing the aspect-oriented paradigm. Business Process Management Journal 16(4), 662–687 (2010)
12. Charfi, A., Mezini, M.: Aspect-oriented web service composition with AO4BPEL. In: (LJ) Zhang, L.-J., Jeckle, M. (eds.) ECOWS 2004. LNCS, vol. 3250, pp. 168–182. Springer, Heidelberg (2004)
13. Costello, C., Molloy, O.: Building a process performance model for business activity monitoring. In: Information Systems Development, pp. 237–248 (2009)
14. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in bpmn. Information and Software Technology 50(12), 1281–1294 (2008)

15. Goedertier, S., Vanthienen, J.: Designing compliant business processes with obligations and permissions. In: Eder, J., Dustdar, S. (eds.) BPM 2006 Workshops. LNCS, vol. 4103, pp. 5–14. Springer, Heidelberg (2006)
16. Hallerbach, A., Bauer, T., Reichert, M.: Managing process variants in the process lifecycle. In: 10th Int'l Conf. on Enterprise Information Systems, ICEIS 2008, pp. 154–161 (June 2008)
17. Jalali, A., Wohed, P., Ouyang, C.: Aspect oriented business process modelling with precedence. In: Mendling, J., Weidlich, M. (eds.) BPMN 2012. LNBIP, vol. 125, pp. 23–37. Springer, Heidelberg (2012)
18. Jalali, A., Wohed, P., Ouyang, C.: Operational semantics of aspects in business process management. In: Herrero, P., Panetto, H., Meersman, R., Dillon, T. (eds.) OTM-WS 2012. LNCS, vol. 7567, pp. 649–653. Springer, Heidelberg (2012)
19. Kang, J.G., Han, K.H.: A business activity monitoring system supporting real-time business performance management. In: Third International Conference on Convergence and Hybrid Information Technology, ICCIT 2008, vol. 1, pp. 473–478 (November 2008)
20. Lanz, A., Weber, B., Reichert, M.: Workflow time patterns for process-aware information systems. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Ukor, R. (eds.) BPMDS 2010 and EMMSAD 2010. LNBIP, vol. 50, pp. 94–107. Springer, Heidelberg (2010)
21. Mulyar, N.A., Schonenberg, M.H., Mans, van der Aalst, W.M.P.: Towards a Taxonomy of Process Flexibility (Extended Version). BPM Center Report BPM-07-11. BPMcenter.org (2007)
22. Pedrinaci, C., Domingue, J., Alves de Medeiros, A.K.: A core ontology for business process analysis. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 49–64. Springer, Heidelberg (2008)
23. Rogge-Solti, A., Weske, M.: Enabling probabilistic process monitoring in non-automated environments. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Wrycza, S. (eds.) EMMSAD 2012 and BPMDS 2012. LNBIP, vol. 113, pp. 226–240. Springer, Heidelberg (2012)
24. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Exception handling patterns. Process-Aware Information Systems. Technical report, BPM Center Report BPM-06-04. BPMcenter. org (2006)
25. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Workflow resource patterns: Identification, representation and tool support. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 216–232. Springer, Heidelberg (2005)
26. Sadiq, S., Governatori, G., Namiri, K.: Modeling control objectives for business process compliance. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 149–164. Springer, Heidelberg (2007)
27. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M.T., van der Aalst, W.M.P.: The proM framework: A new era in process mining tool support. In: Ciardo, G., Darondeau, P. (eds.) ICATPN 2005. LNCS, vol. 3536, pp. 444–454. Springer, Heidelberg (2005)
28. Weske, M., van der Aalst, W.M.P., Verbeek, H.M.W.: Advances in business process management. Data and Knowledge Engineering 50(1), 1–8 (2004)
29. Wohed, P., Russell, N., ter Hofstede, A.H.M., Andersson, B., van der Aalst, W.M.P.: Patterns-based evaluation of open source bpm systems: The cases of jbpm, openwfe, and enhydra shark. Information and Software Technology 51(8), 1187–1216 (2009)