

Proactive Business Process Compliance Monitoring with Event-Based Systems

Robert Thullner
Secure Business Austria
Vienna, Austria
rthullner@sba-research.org

Szabolcs Rozsnyai
IBM T J Watson Research Center
Hawthorne, NY, USA
srozsny@us.ibm.com

Josef Schiefer, Hannes Obweiger, Martin Suntinger
UC4 Software
Vienna, Austria
{josef.schiefer, hannes.obweiger, martin.suntinger}@uc4.com

Abstract—Business processes spanning across organizational boundaries inside and outside an enterprise are increasingly becoming common practice in today's networked business environments. Service level agreements (SLAs) are negotiated between enterprises to measure, ensure and enforce service fulfillment and quality in this dynamic context. Often, SLA violations are directly associated with penalty costs, making it crucial to stick to agreed SLAs and proactively intervene in case of potential violations. Thus, a framework is required which allows for (1) efficient business process compliance monitoring, and (2) taking immediate action in case of compliance violations in order to minimize the business impact. In this paper we present a novel compliance monitoring framework based on a Complex Event Processing (CEP) engine. It allows modeling business processes as event flows, whereby events reflect state changes in a process or the business environment. Compliance checkpoints are added to an event flow and signify aspects which may be relevant to monitor, such as the relative timeframe between two events. Upon these, monitoring rules are defined to detect compliance violations and automatically trigger corrective actions.

Keywords: *Business Process Management, Compliance Monitoring, SLA Monitoring, Complex Event Processing*

I. INTRODUCTION

Increasing legal requirements and growing directives in large organizations require techniques to monitor business processes for compliance policies and react proactively on potential violations to reduce risks of penalties and improve the overall business performance. Recently, this has become a major problem as processes are becoming more complex, spanning across organizational boundaries, while at the same time they are demanding flexibility to ensure the company's competitiveness.

In addition, compliance monitoring requires knowledge about processes and their supporting systems in order to implement precise control points. This can become a major issue, as often the defined process models differ from their

concrete executions. This is especially true in areas that rely on ad-hoc processes with many human-centric interactions. In contrast to highly standardized processes, such as for instance Information Technology Infrastructure Library (ITIL) processes, they require human reasoning and flexibility to solve problems and handle exceptional situations. For instance, in healthcare physicians need degrees of freedom to diagnose and initiate treatments, while at the same time regulations require to perform standardized actions to ensure quality standards.

The systems forming the pillars of such processes are also steadily growing in size as well as complexity as they are becoming more federated and loosely coupled. As these systems and applications become larger and more interconnected, the efforts of proactively monitoring compliance becomes a major issue.

In this paper, we introduce a compliance monitoring framework based on the Complex Event Processing (CEP) [1][7] paradigm, which allows investigating data flowing through organizations in real time. The framework enables organizations to automate the monitoring of business process and proactively reduce compliance violations.

The proposed compliance framework is an extension of the event-based system SARI (Sense and Response Infrastructure) [9][15], which is a CEP system that allows users to compose complex pattern-detection logic [10] [16] [17] by visually modeling event-triggered decision graphs.

The left side of Fig. 1 shows an overview of SARI's high level architecture and the extensions for compliance monitoring. In SARI, events are received from and propagated to various event producers and consumers. Incoming events can be correlated based on event attributes and therefore it is possible to identify which events belong to the same instance of a business process [14]. The business logic of a SARI application is implemented in several event services, where each service encapsulates a part of the overall business logic. Incoming events are passed to event services that process them according to their business logic. An event service can republish a received event or publish

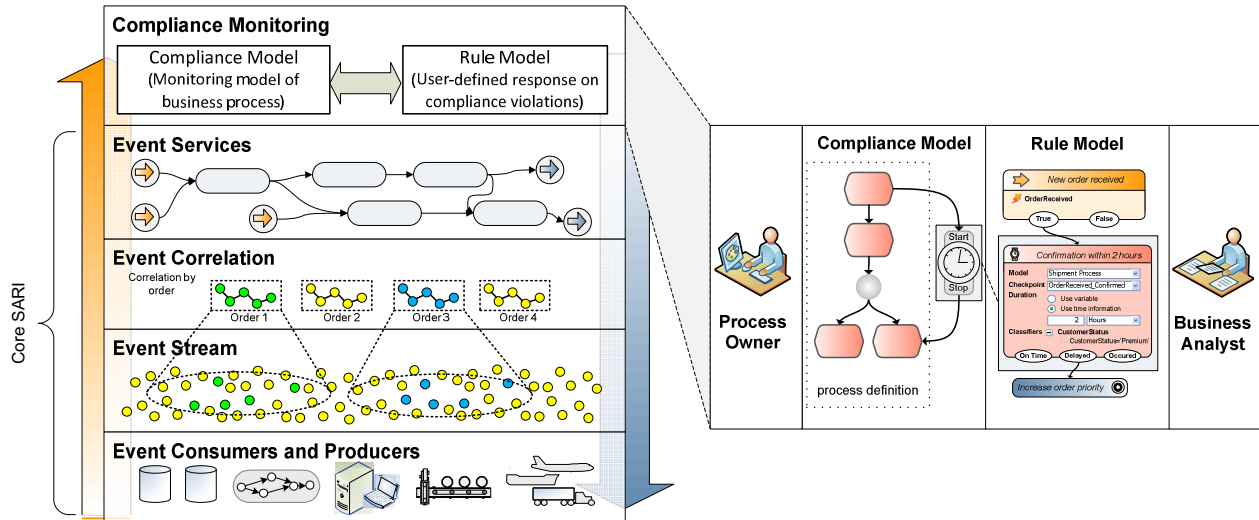


Figure 1. Business process compliance monitoring with SARI

new events. Event services are orchestrated, to build the complete business logic of an SARI application. For example, an event service can be used to enrich incoming events with additional information. The enriched events are passed on to subsequent event services which can execute further business logic on the events, like evaluating rules on events. On the top of SARI, a compliance model defines possible event flows of a business process and compliance checkpoints for that process. When receiving events, the SARI system evaluates the checkpoints and uses user-defined rules for responding with appropriate actions. For instance, if SARI detects a process pattern that does not comply with a defined policy, a response action is emitted which could be the triggering of an alarm.

Our compliance monitoring framework consists of two parts: 1) a compliance model, and 2) a rule model. An illustration of the framework is shown on the right side of Fig. 1. We use the following nomenclature for referring to elements of our framework. We use the term *business process* to refer to a general process of an enterprise without any special description of the process. A *process definition* is a business process modeled by an event flow in our compliance framework. A *compliance model* is a process definition enhanced by compliance checkpoints which are used for monitoring the business process.

Compliance models are used to define a monitoring perspective of business processes described by events and event flows. They consist of a process definition and compliance checkpoints in order to monitor the process. Compliance models are usually built by the process owner of a business process. The rule model is primarily used by business analysts for detecting business situations of an enterprise and triggering appropriate actions according to the detected situation. For example in the job scheduling domain, rules can be applied to detect if too many jobs ended abnormally on a server and trigger an automatic message to the administrator. We enhanced the rule model of [16] with new components which refer to checkpoints in compliance

models. For example, if a support request from a premium customer is not answered within two hours, an escalation of the support request is automatically performed. Creating the process definition and inserting compliance checkpoints is part of the compliance model. The definition of the parameters for the checkpoints (e.g. premium customer, response in two hours) as well as the response actions for a process compliance violation is specified within a rule. By splitting the framework into a compliance model part and a rule model part, we achieve a separation of concerns and reusability of individual models for different business purposes.

In this paper, we focus on the compliance model and discuss the extensions of the SARI rule model. Technical event processing issues, like handling of time, synchronization, out-of-order events, etc. are not covered in this paper. These issues are covered by the underlying CEP system SARI and are discussed in [14][15][16][17].

The remainder of this paper is organized as follows. In section II, we give an overview about related work. The compliance model of our framework is introduced in section III. In section IV, we discuss the rule model and its enhancements for monitoring compliance checkpoints. A real-world use case is discussed in section V which demonstrates how our framework can be used for monitoring a shipment process. Section VI gives an overview of the runtime architecture of the framework. Finally, a conclusion and outlook for future work are given in section VI.

II. RELATED WORK

Compliance monitoring, policy or contract enforcement has been a subject of interest of several research efforts. Each of them deals with specific aspects of this problem domain by choosing different approaches to solve them. In this section we want to outline, relevant contributions with the aim to provide a distinction to our work and sharpen the understanding of our own contributions.

Cubera et al. [4] identified three features of a compliance solution. Those are support for root cause analysis, continuous monitoring and proactive prevention of compliance violations. Their approach for business provenance provides a generic data model and middleware infrastructure to collect and correlate information about how data was produced, what resources were involved and which tasks were executed.

Various approaches focus on event-centric perspectives to collect and present compliance or contractual violations without actively reacting on them [9][20]. However, others such as [6] monitor SLAs to activate contractual norms such as when a storage spaces exceeds the limits a payment obligation is activated. This allows users to query for active or non-active norms.

Rozinat et al. [12] propose an incremental approach to check the conformance of a process model and an event log. By performing a conformance analysis, their approach aims at the detection of inconsistencies between a process model and its corresponding execution log, and their quantification by the formation of metrics.

Businesses that need to deal with large and complex policies, laws or regulations require means of expressing complex compliance situations and patterns. Event-Condition-Action rules (ECA) [16][18][19] allow to model policies that are activated in case that certain events or patterns of events occur. If a situation has been detected a response (i.e. action) is triggered [2][19].

Event-Based systems have shown to be a suitable paradigm to enable a proactive compliance monitoring [3][5][11][20]. SARI (Sense and Respond Infrastructure) [9][15] is an event-based solution which allows to model business situations and exceptions with event-driven rules that are designed to be created and modified by business users.

Mulo et al. [9] propose an approach for monitoring business compliance in service-oriented systems. In their approach, a service invocation is represented as an event, enabling them to map business process activities into trails of events that make up compliant business processes. The event trails guide the creation of rules, which are leveraged by CEP techniques to monitor business processes for compliance.

Similar to concepts introduced in this paper Schumm et al. [21] introduced a concept in context of the Compass project [22] which encapsulates compliance check formulas into reusable process fragment. These formulas in the compliance fragments consist of rules which can be verified against process models at design time to reduce integration efforts.

Middelton et al. [8] introduce a health-care domain centric compliance solution that is based on declarative policies. It is notable that their scenario is based on ad-hoc activities which make it a hard problem to put mechanisms in place that ensures that certain policies are violated or are in danger to be violated as there is no process model in place. Instead their monitoring framework utilizes event-listing agents based on policy databases that are activated when certain conditions on events are met. Our approach also separates the knowledge space of compliance from its

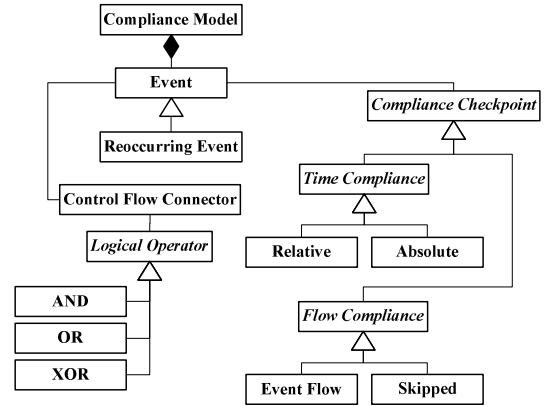


Figure 2. Meta-model of compliance model

application layer such as in rules. However, our proposed event-driven rule model and event processing capabilities allow expressing richer event patterns for general purpose usage.

Compliance checking between business processes and business contracts was already discussed by Governatori et al in [23]. In this work business contracts are analyzed and expressed by a formal language. The derived language constructs are then applied to defined business process (which are defined in BPMN notation) to identify inconsistencies between a business contract and a defined business process. In contrast to this work, our goal is not to identify deviations between business contracts and business processes, but to enable continuous, real-time monitoring of business process executions against user-defined values. Furthermore we want to enable immediate reactions to violations of defined values.

In contrast to some previously mentioned works, our introduced even-based framework pays special attention to continuous monitoring across system boundaries (i.e. distributed processes executions). Several event adapters are available which collect events from various systems of an enterprise in real time. The collected events are continuously processed by compliance and a rule model to monitor business processes and react to compliance violations. Immediate reactions to business process violations are defined in the rule model of our compliance framework. In order to proactively prevent process violations some event forecasting functionality has to be included in our framework which is key part of our future work.

III. COMPLIANCE MODEL

The first part of our monitoring framework is the compliance model. The purpose of this model is to define business process represented by event flows (process definition) and insert compliance checkpoints to be able to monitor the process. For the compliance model, it is assumed that the events which are used for process definitions arrive in order. To guarantee this assumption, pre-processing components have to be deployed which are able to sort events, so that out-of-order events are sorted correctly again, before they are forwarded to the compliance model.

A. Process definition

The process definition describes business processes, represented as flows of business events. In our model event flows typically include business level events. For simplicity reasons, we assume that the business level events have been aggregated or derived from low level “technical” events if they have not been already provided in their abstracted form. Usually, events relating to a business process mark a significant point in time such as the start and/or the end of an activity in a business process. When looking at the supply chain management domain, typical events would be for instance “*OrderReceived*” when an order was received from a customer or “*ShipmentStarted*” when the shipment of an order started.

The meta-model of the compliance model is an adapted version of the event-driven process chains (EPC) model [13]. For creating process definitions we reuse events and connections from EPC. For monitoring a business process, we extend the model by compliance checkpoints. The complete meta-model is shown in Fig. 2.

A process definition consists of two or more events, which are connected by control flow connectors. Control flow connectors can be further connected to logical operators. The semantic meaning of a connection is a “*has to occur after*” relationship. When event X and event Y are connected, it means that event Y has to occur after event X in the process. We introduce a specialized event type, referred to as *Reoccurring Event*. Events can be marked as *reoccurring*, which means that this event can occur any number of times at a particular stage of a process definition, replacing the need for a loop that spans across a single event. To construct a complete compliance model, the events of a process definition can further be connected to compliance checkpoints. These are divided into time and flow compliance checkpoints, where each of them is further divided into specialized checkpoints.

The visual representation for building process definitions is also reused from the event-driven process chain (EPC) notation. By reusing this visualization, users who are already familiar with EPC can easily understand the compliance model. Additionally, existing EPC models can be converted to compliance models by just considering events and removing all other EPC artifacts. Thus existing EPC process definitions can easily be transformed into compliance models and reused for compliance monitoring.

B. Compliance checkpoints

Compliance checkpoints are added to a process definition to form a complete compliance model. Monitoring is supported by two types of compliance checkpoints:

- *Time compliance*: to monitor if the execution of a task in a process occurred within a specified timeframe. For example, a customer order has to be confirmed within two hours.
- *Flow compliance*: to monitor if the order of executed tasks is compliant to the defined order of tasks in the process definition. For example an invoice to a customer must not be sent before a product is

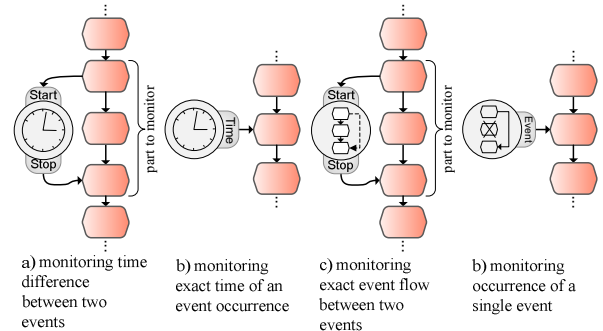


Figure 3. Compliance checkpoints

delivered. Otherwise a customer might be charged for products which have not been delivered yet.

Within a compliance model, a checkpoint is used to identify events and event flow segments of a process that are relevant for compliance checking. A checkpoint has an identifier and may have an arbitrary number of *classifiers*. The identifier allows rules to refer to a checkpoint within a compliance model. Classifiers allow creating conditions for distinguishing different cases under which the checkpoint should be evaluated. A classifier is defined by an identifier and a mapping to an event attribute of the event to which a checkpoint is connected. The identifier of the classifier can be used within rules to create conditions on the mapped event attribute. For example, a classifier *CustomerStatus* is mapped to the customer status field of an “*OrderReceived*” event. Now a condition like “*CustomerStatus='Premium' OR CustomerStatus='Gold'*” can be defined within a rule to limit the monitoring only to premium or gold customers. If multiple classifiers are defined for a checkpoint, the user can decide if the conditions defined within rules are concatenated by an AND or an OR operation. Thus, by using classifiers a business analyst can restrict the monitoring to specific process instances of a business process only.

Time Compliance. Time compliance is monitored in two ways: 1) a *relative timing checkpoint* and 2) an *absolute timing checkpoint*. Time compliance checkpoints are defined with parameters for the timing information in the compliance model, whereby the actual values for the parameters are specified by business users in rules. A checkpoint may be used in several rules with different parameter settings defined by a user. For example, the monitoring requirements for a premium customer might be different than the requirements for normal customers.

The *relative timing checkpoint* (Fig. 3a) monitors the time between the occurrences of two events. It has a start and an end port which both can be connected to one or more events of the process definition, where the events connected to the start port have to differ from the events connected to the stop port. For the connected events the difference in time is monitored. The typical usage scenario of this checkpoint is monitoring the maximum allowed duration for a business activity, for instance checking whether an order confirmation was sent on time. An exception, where it is allowed to connect the start and the end port to the same event are

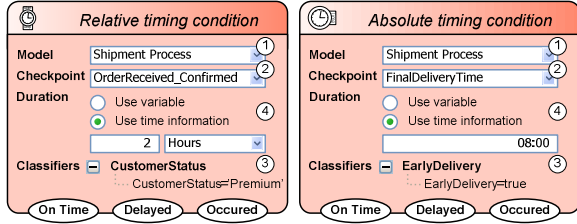


Figure 4. Rule components for monitoring time compliance checkpoints

events marked as *reoccurring*. In this case, the checkpoint monitors the time difference between the occurrences of two events of the same type.

The *absolute timing checkpoint* (Fig. 3b) monitors the time of occurrence of a single event. Therefore, it may be connected to a single event only. In contrast to the relative timing checkpoint (where time periods are specified), an exact time to monitor has to be specified for this checkpoint. The time is always monitored relative to the day of occurrence of the predecessor event in the process definition. The major usage for this checkpoint is that an activity has to start or has to be finished at a fixed point in time, regardless of any previous activities. For example, a morning delivery of a product to a customer has to be completed before eight o'clock in the morning. If the delivery is done later, the checkpoint is violated.

Flow Compliance. Flow compliance is also supported by two checkpoints: 1) an *event flow checkpoint* and 2) an *event skipped checkpoint*.

The *event flow checkpoint* (Fig. 3c) monitors if the event flow occurs exactly as designed in the process definition. The start and end port of the checkpoint are connected to the first resp. last event(s) of the part of the process definition for which the flow compliance shall be monitored. If the events occur in different order as designed a violation is detected. Typical usages for such checkpoint are processes where the exact task order is required for legal or other compliance reasons.

A special form of the event flow checkpoint is the *event skipped checkpoint* (Fig. 3d). This checkpoint is connected to one event only and monitors if an event was skipped during the process execution. The difference to the event flow checkpoint is that this checkpoint only monitors a single event for a special situation, whereas event flow checkpoints monitor situations of an arbitrary event set.

We show the usage of all compliance checkpoints in an example scenario in section V.

IV. RULE MODEL

The second part of our compliance monitoring framework is the rule model. The rule model is used to detect event patterns which describe business situations. For a defined business situation, the user may associate actions which should be performed automatically once the situation is detected. The foundation of the rule model is described in [16]. We extended the rule model with new rule components for testing the checkpoints of compliance models.

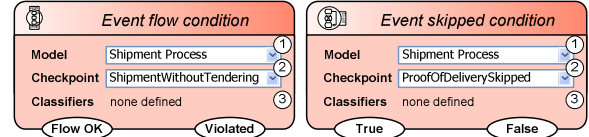


Figure 5. Rule components for monitoring flow compliance checkpoints

As described in the previous section, it is possible to use a compliance model for monitoring purposes in different rules, because compliance checkpoints may have parameters which can be specified by users within the rules. At runtime, the rule parameters are passed to the compliance model. For example, the time between an order is received until it is confirmed might differ depending on the customer type. For a premium customer, a confirmation might already be required after two hours whereas for normal customers, a confirmation within four hours is still acceptable. This information is specified in a rule and is passed as parameter to the compliance model.

We propose a new rule component for each compliance checkpoint (a *relative timing condition*, an *absolute timing condition*, an *event flow condition* and an *event skipped condition*). Fig. 4 and Fig. 5 show the rule components for monitoring compliance checkpoints.

All rule components specify the compliance model (1) and the monitored checkpoint (2). At the bottom of each component the classifier conditions (3) of the compliance checkpoint are defined. In the relative timing condition in Fig. 4 a condition for the customer status is set. This causes the output ports to be triggered only for premium customers. Classifier conditions are optional input fields. In case that none are specified, all orders are monitored.

Either a fixed value or a variable value is used to monitor a time compliance checkpoint (4). This value defines the runtime information for the checkpoint. A variable value can be passed as input parameter to a rule which is further forwarded to the compliance model. Both timing conditions contain three ports which can be connected with other rule components.

The “*on time*” port of the relative timing condition is triggered when the stop event of the checkpoint occurred within the provided timeframe. For the absolute timing condition, the “*on time*” port is triggered if the event of the checkpoint occurred before the specified time value. For both, the “*delayed*” port is triggered in case that the connected event did not occur on time. Finally, the “*occurred*” port is triggered when the event occurred, independent whether it was on time or not.

For flow conditions, the “*flow ok*” port of the event flow condition is triggered if no violation was detected whereas the “*violated*” port is triggered otherwise. With skipped event conditions, “*true*” and “*false*” ports indicate whether the associated event was skipped or not.

V. EXAMPLE: MONITORING OF A SHIPMENT PROCES

In the following we use a real-world scenario of an end-to-end shipment business process from the supply chain

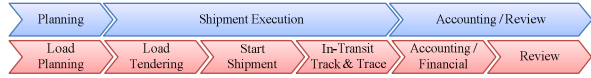


Figure 6. High-level shipment process

TABLE I. EVENTS OF SHIPMENT PROCESS

Event	Description
OrderReceived	An order was received from a customer.
OrderConfirmed	A received order was confirmed
ShipmentTenderIssued	A shipment tender was sent to a carrier.
TenderRejected	A carrier rejected a tender.
TenderAccepted	A carrier accepted a tender.
TransportationStarted	Transportation of product started.
LocationUpdated	Current location of delivery was updated.
TransportationFinished	Transportation of production finished.
ProofOfDeliveryReceived	A proof of delivery was received.
InvoiceSent	An invoice was sent to the customer.
ShipmentReviewed	A shipment was reviewed.

management domain to show how our framework models interact. The shipment of products is divided into three main phases: 1) *planning*, 2) *shipment execution* and 3) *accounting and review*. The high-level business process is shown in Fig. 6. When orders are received from customers an order management system plans the delivery of the ordered items to the customer. The result of the planning activity is the selection of a preferred carrier to deliver the ordered products. The shipment execution phase starts with asking the selected carrier for delivering the products by sending a shipment tender. If the carrier denies the tender, another carrier from the planning phase is asked and so forth. When a tender is accepted, the carrier picks up the products from the manufacturer and delivers them to the customer. During delivery the carrier sends periodic updates about the delivery

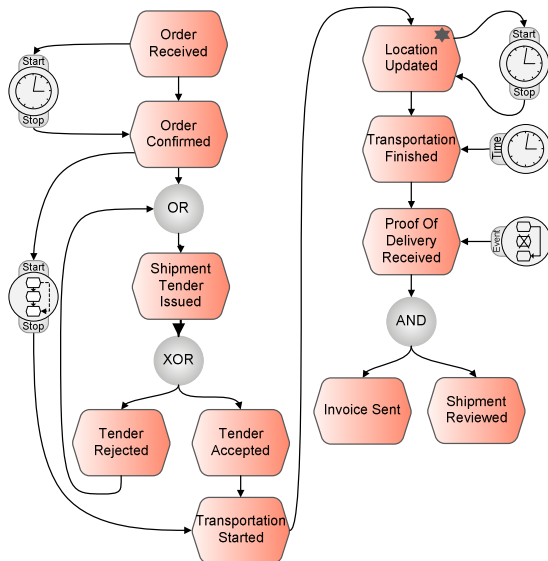


Figure 7. Compliance model for shipment process

status to the manufacturer. The final phase of the process is to review the shipment.

The first step in building a compliance model is to identify the relevant events of the business process. We use 11 events to represent the most important state changes of the shipment process. All events with a short description are listed in table I. Once all events of the business process are identified, the next step is to create the process definition by structuring the events to an event flow which reflects the activities of the business process. After the process definition is completed, compliance checkpoints are added to finalize the compliance model.

The compliance model of the shipment process is shown in Fig. 7. We use five checkpoints to monitor the process. A relative timing checkpoint monitors the time between an order is received and its confirmation. Another relative timing checkpoint monitors the time between two “*LocationUpdated*” events. This event is marked as *reoccurring* (indicated by the star at the top right of the event). Therefore it is possible to connect the start and end port of the relative timing checkpoint to the same event. An event flow checkpoint between the “*OrderConfirmed*” event and the “*TransportationStarted*” event is used to ensure that a tendering process is fully compliant to the designed event flow. An absolute timing checkpoint is inserted at “*TransportationFinished*” event to be able to monitor the final delivery time to the customer, in case some special delivery options (e.g. morning deliveries) were negotiated. A final skipped event checkpoint is inserted at the “*ProofOfDeliveryReceived*” event to be able to recognize if a customer was invoiced without any received delivery.

Once the compliance model is finished, rules can be created that include compliance monitoring information. We show two exemplary rules in Fig. 8. The first rule increases the order priority of a received order whenever it was not confirmed within two hours for a premium customer. The second rule monitors the event skipped condition on the “*ProofOfDeliveryReceived*” event. When the event is skipped during the process execution, the process owner as well as the customer account manager is informed.

In order to be able to monitor business environments a SARI application has to be created which includes the compliance model as well as monitoring rules. The

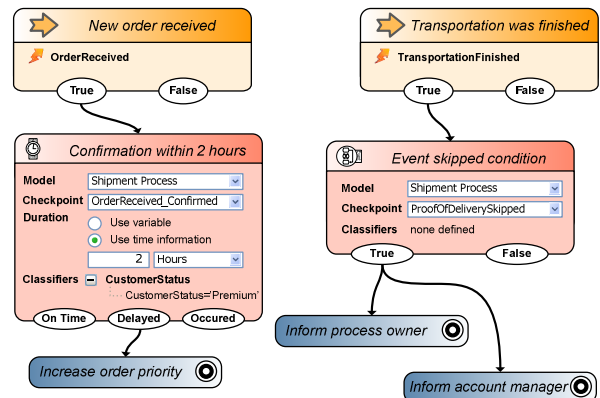


Figure 8. Rules for monitoring shipment process

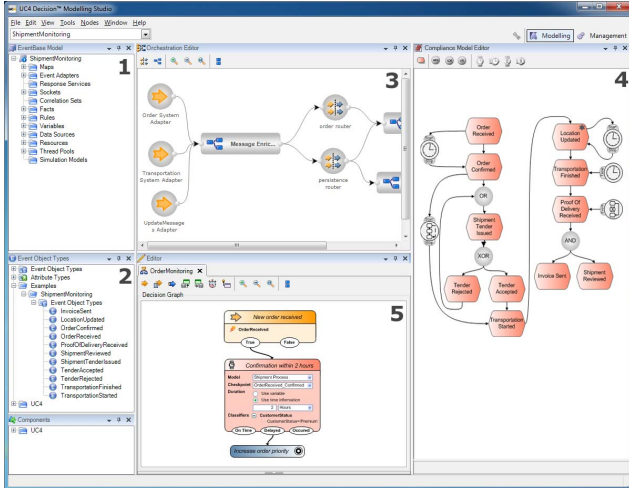


Figure 9. Application Modelling Editor

compliance model and rules are built using a graphical editor for fast creation of artifacts. Generally, a SARI application not only consists of the compliance model and rules, but also includes all relevant artifacts, which are required for application execution and event monitoring. For example, a complete SARI application consists of event adapters and response adapters used for collecting events from event producers and emitting event to event consumers. Furthermore correlation information between types of events, like correlating the order identifier of an “*OrderReceived*” and an “*OrderConfirmed*” event is defined in a SARI application. For application execution, thread pools can be defined and assigned to components of the application. By assigning thread pools it is possible that components processing heavy loads have more threads available than components only processing insignificant load. Consequently a SARI application can be tuned by well-thought-out assignment of thread pools to application components.

Fig. 9 shows the graphical editor for building SARI applications. In part 1 of the figure all components (event adapters, correlation information, rules, etc.) of an application are listed. In part 2, event objects and their attributes are defined. One can find all events used for our example listed there. An editor for orchestrating event services and defining the business logic of a SARI application is shown in part 3. It can be seen that events are read from three event sources and forwarded to an enrichment service. After the processing of events in the enrichment service, they are sent to the next processing components, like rule services which execute the rule logic. Finally in part 4 and 5 of the figure, the graphical editors for building compliance models on the one side and rules on the other side are shown.

We have chosen to design and use a graphical editor for building compliance and rules models and not to use any formal language definition to make the use of the framework easily accessible for any kind of users. A graphical drag-and-drop style of building compliance models is usually easier to learn, understand and intuitive to use for end-users compared

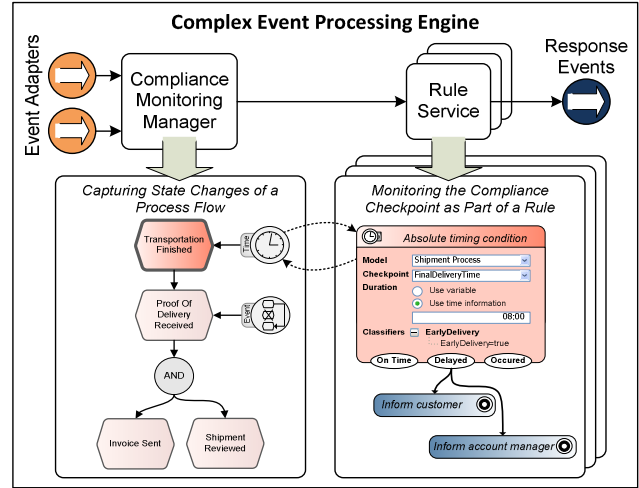


Figure 10. Runtime architecture

to learning a new formal language to build compliance models. The interface was tested with end users, which are typically system operators with a sound technical understanding but no in-depth know-how and showed that the user interface was well received and easy to use.

The example used in this scenario monitored a single process instance against values directly defined in rules. In real time environments users are also interested in collected statistics on process executions and use the statistical values to monitor future process executions. For example a typical scenario could be that a user is not interested if an order is confirmed within two hours (like shown in Fig. 8) but is interested if the order confirmation is takes 50% longer that the average order confirmation response time. For covering such requirements, SARI offers a module, called *Scoring*, which is capable to store any statistical values which can be extracted and calculated from incoming events. The values can then be used in rules to be able to monitor process executions based on statistical values.

VI. RUNTIME ARCHITECTURE

A SARI application is executed inside SARI’s CEP engine. The engine consists of one administration node and several worker nodes. A SARI application can be deployed on one or more workers which collect events from event sources, execute the monitoring and rule logic, and emit events to event consumers. The administration node is responsible for orchestration and mediation between workers.

The runtime architecture of the compliance monitoring part of SARI is shown in Fig. 10. As discussed in the introduction, the business logic of a SARI application is separated into several event services, each executing a part of the business logic. This principle is followed by the rule model of our framework. Rules are executed inside special event services, so-called *Rule Services*. One rule can be executed in more than one rule services so that a rule can be reused at the different stages of a SARI application. The rule

ervice is responsible for evaluating rules and triggering the response actions defined within rules.

The component responsible for managing and evaluating all instances of compliance models is the *compliance monitoring manager*. In opposite to rules, this component is not executed within an event service, but is globally available in SARI on the administration node. The reason for this is that compliance models can be reused in several rules and therefore a global entity is required. All events received from source systems via event adapters are forwarded to the compliance monitoring manager. Whenever the first event of a compliance model is received, an instance of the compliance model is created. For all subsequent events of the process definition the correlation information ensures that the correct instance of the compliance model processes the event. When the compliance monitoring manager finished processing an event, it is forwarded to the further event services, like the rule service, that execute business logic on incoming events.

Rules services communicate with the compliance monitoring manager by a publish-and-subscribe mechanism. Whenever a rule with compliance monitoring conditions is initialized the worker on which the rule service is executed subscribes to the compliance monitoring manager at the administration node for receiving monitoring information from checkpoints of the referred compliance model. When events are received by the compliance monitoring manager, it evaluates the checkpoints which are connected to the event. If subscriptions for that checkpoint exist it will notify the subscribed worker to trigger the corresponding port of the compliance monitoring condition inside a rule.

VII. CONCLUSION AND FUTURE WORK

In this paper, we presented a framework for monitoring the compliance of business processes using the Complex Event Processing paradigm. Our framework uses the SARI CEP engine for monitoring intra- and inter-organizational business processes.

The framework offers high flexibility to process owners when creating compliance models. On the one hand, a process owner can create a process definition based on the most relevant events of a business process only. This approach leaves a degree of freedom for ad-hoc activities within a business process because only the most import events are included in the compliance model and detailed events are not monitored. On the other hand, also compliance models for standardized business processes can be created by including all events of the business process in the process definition, which leaves no room for ad-hoc activities within a business process.

We proposed a compliance model which can be fully built by a graphical editor without the need for much technical or programming skill by end-users. We adapted the well-known EPC meta-model for creating compliance models. This facilitates organizations to easily transform existing EPC process definitions into compliance models.

As our framework separates the definition of business processes from the compliance monitoring requirements we can build reusable compliance models that can be used in

independent rules for different business purposes. By combining the compliance model with the rule model, we are able to monitor business process and immediately react to compliance violations of a process. This is major advantage in comparison to post-execution analysis tools. When reactions on process violations are performed immediately after the violation is detected, an enterprise can avoid or at least minimize penalty fees for SLAs. Also operations cost of an enterprise can be reduced as immediate corrections to violations are usually cheaper to achieve as corrections after a post-analysis of process compliance where the given business situation already changed and corrections might require further additional corrections in dependent business processes.

The framework presented in this paper is part of a long term research in the field of Complex Event Processing with focus on compliance and SLA monitoring and forecasting for event-based systems. A special focus will be to include forecasting capabilities into the compliance model. By being able to forecast an event in the event flow, corrective actions can even be taken before critical situations arise. Currently event prediction usually focuses on forecasting special events, like failure events. In our case we want to be able to predict the occurrence of any arbitrary event. Solving this prediction problem will be a major part of our future research. Another part of future research will focus on dealing with inter-organizational issues like security and trust, when performing compliance monitoring on business processes, spanning across organizational boundaries.

REFERENCES

- [1] D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik, "Aurora: a new model and architecture for data stream management," *The VLDB Journal*, vol. 12, pp. 120–139, August 2003.
- [2] J. Bailey, A. Poulouvasilis, and P. T. Wood, "An event-condition-action language for xml," in *Proceedings of the 11th international conference on World Wide Web*, ser. WWW '02. New York, NY, USA: ACM, 2002, pp. 486–495.
- [3] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf, "Design and evaluation of a wide-area event notification service," *ACM Trans. Comput. Syst.*, vol. 19, pp. 332–383, August 2001.
- [4] F. Curbera, Y. Doganata, A. Martens, N. K. Mukhi, and A. Slominski, "Business provenance — a technology to increase traceability of end-to-end operations," in *On the Move to Meaningful Internet Systems: OTM 2008*, ser. Lecture Notes in Computer Science, vol. 5331. Springer Berlin / Heidelberg, 2008, pp. 100–119.
- [5] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surv.*, vol. 35, pp. 114–131, June 2003.
- [6] A. D. H. Farrell, M. J. Sergot, M. Salle, and C. Bartolini, "Using the event calculus for the performance monitoring of service-level agreements for utility computing," in *In Proceedings of First IEEE International Workshop on Electronic Contracting (WEC 2004)*, 2004.
- [7] D. C. Luckham, *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001.
- [8] G. Middleton, L. Peyton, C. Kuziemy, and B. Eze, "A framework for continuous compliance monitoring of health processes," in *Proceedings of the 2009 World Congress on Privacy, Security, Trust and the Management of e-Business*, ser. CONGRESS '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 152–160.

- [9] E. Mulo, U. Zdun, and S. Dustdar, "Monitoring web service event trails for business compliance," in *Service-Oriented Computing and Applications (SOCA), 2009 IEEE International Conference on*, 2009, pp. 1–8.
- [10] H. Obweiger, J. Schiefer, P. Kepplinger, and M. Suntinger, "Discovering hierarchical patterns in event-based systems," in *Proceedings of the 2010 IEEE International Conference on Services Computing*, ser. SCC '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 329–336.
- [11] P. R. Pietzuch, "Hermes: A scalable event-based middleware," Ph.D. dissertation, University of Cambridge, 2004.
- [12] A. Rozinat and W. M. P. van der Aalst, "Conformance checking of processes based on monitoring real behavior," *Inf. Syst.*, vol. 33, pp. 64–95, March 2008.
- [13] A.-W. Scheer, *Aris-Business Process Modeling*, 3rd ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2000.
- [14] J. Schiefer and C. McGregor, "Correlating events for monitoring business processes," in *ICEIS (1)*, 2004, pp. 320–327.
- [15] J. Schiefer and A. Seufert, "Management and controlling of time-sensitive business processes with sense & respond," in *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce Vol-1 (CIMCA-IAWTIC'06) - Volume 01*, ser. CIMCA '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 77–82.
- [16] J. Schiefer, S. Rozsnyai, C. Rauscher, and G. Saurer, "Event-driven rules for sensing and responding to business situations," in *Proceedings of the 2007 inaugural international conference on Distributed event-based systems*, ser. DEBS '07. New York, NY, USA: ACM, 2007, pp. 198–205.
- [17] J. Schiefer, H. Obweiger, and M. Suntinger, "Correlating business events for event-triggered rules," in *Proceedings of the 2009 International Symposium on Rule Interchange and Applications*, ser. RuleML '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 67–81.
- [18] M. Seiriö and M. Berndtsson, "Design and implementation of an eca rule markup language," in *Rules and Rule Markup Languages for the Semantic Web*, ser. Lecture Notes in Computer Science, A. Adi, S. Stoutenburg, and S. Tabet, Eds. Springer Berlin / Heidelberg, 2005, vol. 3791, pp. 98–112.
- [19] C. S. Shankar, A. Ranganathan, and R. Campbell, "An eca-p policy-based framework for managing ubiquitous computing environments" in *Proceedings of the The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 33–44.
- [20] M. Strano, C. Molina-Jiménez, and S. K. Shrivastava, "Implementing a rule-based contract compliance checker," in *I3E*, 2009, pp. 96–111.
- [21] D. Schumm, F. Leymann, Z. Ma, T. Scheibler, and S. Strauch. "Integrating Compliance into Business Processes: Process Fragments as Reusable Compliance Controls", in *Proceedings of the Multikonferenz Wirtschaftsinformatik (MKWT'10)*, Universitaetsverlag Goettingen, 2010.
- [22] "Compas", <http://www.compas-ict.eu>, Web. 26 February 2011.
- [23] G. Governatori, Z. Milosevic, and S. Sadiq, "Compliance checking between business processes and business contracts", in *Proceedings of the 10th International Enterprise Distributed Object Computing Conference (EDOC)*, IEEE Computer Society, 2006, pp. 221-232