Towards a Service-Oriented Architecture for Operational BI

A Framework for Rule-Model Composition

Josef Schiefer[1], *Andreas Seufert*[2]

[1]*UC4 Senactive Software GmbH, Wolfsgraben (Österreich)*
[2]*Institut für Business Intelligence, Steinbeis Hochschule Berlin*

1 Introduction

The information revolution is sweeping through our economy. Dramatic reductions in the cost of obtaining, processing, and transmitting information are changing the way we do business. Information technology is transforming the nature of products, processes, companies, industries and even the competition itself.

As a consequence new approaches have to be put into action for decision-making, too (Clark et al. 2007; Davenport and Harris 2007). One of the key challenges is to move decision making applications (Business Intelligence) out of the back room and embeds it into business, intertwining it with operational processes and applications that drive current daily decisions. In essence, Operational Business Intelligence merges analytical and operational processes into a unified whole (Eckerson 2007; Van der Aalst 2004).

Therefore, information flows from different sources in the form of messages or events, giving a hint of the state at a given time, such as the completion of shipping an order have to be monitored and managed in real time.

Business Process Management (BPM) systems, also named as Business Process Intelligence (BPI) systems, are software solutions that support the management of the lifecycle of a business process (Grigori et al. 2004). In this context SOA concepts and technologies for building BI applications are looked at to be of great importance (Linders 2008). For the execution of business processes, many organizations are increasingly using process engines supporting standard-based process models (such as WSBPEL) to improve the efficiency of their processes and keep the testing independent from specific middleware.

A major challenge of current BPM solutions is to continuously monitor ongoing activities in a business environment (Sayal 2002) and to respond to business events with minimal latency (Ingvaldsen and Gulla 2006). One of the most promising concepts that approaches the problems of closed-loop decision making and the lack of gaining real-time business knowledge is the concept of Complex Event Processing (CEP), which was first introduced by Luckham (2005).

In this paper, we introduce the rule management of the CEP system SARI (Sense And Respond Infrastructure) which was proposed by Schiefer and Seufert (2005). The SARI system is able to process large amounts of events, and provides functions to monitor, steer and optimize business processes in real time. The system automatically discovers and analyzes business situations or exceptions and can create reactive and proactive responses, such as generating early warnings, preventing damage, loss or excessive cost, exploiting time-critical business opportunities, or adapting business systems with minimal latency.

In SARI, business situations and exceptions are modeled with sense and respond rules which have been designed to be created and modified by business users. SARI offers a user-friendly modeling interface for event-triggered rules, which allows to model rules by breaking them down into simple, understandable elements. The main contribution of this paper lies in a new model for constructing rules with a correlation model and a graph for representing business situations with a combination of event conditions and event patterns. The proposed model can be seamlessly integrated into the distributed and service-oriented event processing platform.

The remainder of this paper is organized as follows. In Section 2, we review related work. Section 3 introduces the framework demonstrates a concrete example. In Section 4, we discuss the framework including runtime aspects of the rule processing. Finally, in Section 5 we conclude our paper and give an outlook for future work.

2 Related Work

Related work can be divided into work on active event processing, event algebras in the active database community, work on event/action logics, updates, state processing/transitions, and temporal reasoning in the knowledge representation domain.

There has been a lot of research and development concerning knowledge updates and active rules in the area of active databases and several techniques (Baralis and Widom 1994; Bailey et al. 1997) based on syntactic (e.g. triggering graphs or activation graphs), and semantics analysis of rules have been proposed to ensure termination of active rules (no cycles between rules) and confluence of update programs (always one unique outcome). The combination of deductive and active rules has been also investigated in different approaches mainly based on the simulation of active rules by means of deductive rules (Ludascher 1998). However, in contrast to our work, these approaches often assume a very simplified operational model for active rules without complex events and ECA-related event processing.

Several Complex Event Processing (CEP) and Event Stream Processing (ESP) systems have been developed, where many of them use an SQL-based approach for querying event streams. An example is Esper (2007), which is an Open Source Event Stream engine that allows to analyze event streams with SQL-queries for defining correlations between events and for detecting event patterns. Aurora (Abadi et al. 2003), as well as its successors Borealis (Abadi et al. 2005) and Medusa (Zdonik et al. 2003), are also SQL-based processing engines, which provide efficient scheduling service and QoS delivery mechanisms.

RuleCore, an approach proposed by Seirio and Berndtsson (2005), is an event driven rule processing engine supporting Event Condition Action (ECA) rules, and providing a user interface for rule building and composite event definition.

Wu et al. (1998) propose an event correlation approach with rules in the "conclusion if condition" form which are used to match incoming events often via an inference engine. Based on the results of each test, and the combination of events in the system, the rule-processing engine analyzes data until it reaches a final state.

Chen et al. (2006) show an approach for rule-based event correlation. In their approach, they correlate and adapted complex/structural XML events corresponding to an XML schema. The authors describe an approach for translating hierarchical structured events into an event model which uses name-value pairs for storing event attributes.

ECA rules have been also proposed by several authors for workflow execution, e.g., Barbará et al. (1994), Bussler and Jablonski (1994), and Dayal et al. (1990). In event-driven workflow execution, events and event-condition-action rules are the fundamental mechanisms for defining and enforcing workflow logic. Processing entities enact workflows by reacting to and generating new events. The foundation on events facilitates the integration of processing entities into coherent systems. Some of these systems (Barbará et al. 1994; Dayal et al. 1990) use composite events to detect complex workflow situations. EVE is a system proposed by Geppert and Tombros (1998) using ECA rules for workflow management addressing the problem of distributed event-based workflow execution.

3 Framework for Event-Based Rule Services

Sense and respond rules allow detecting business situations by discovering event patterns. When an event patterns has been discovered, they can automatically trigger responses. The key requirements of sense and respond rules can be summarized as follows:

Event-triggered Rule Evaluation. Sense and respond rules enable companies to monitor their business, IT and organizational processes in real time, and respond to exceptions and capitalize on time-sensitive business opportunities as soon as new events occur within the business environment. In other words, the evaluation of sense and respond rules is triggered by events delivering the most recent state and information from the business environment.

User-friendly Rule Modeling. Sense and respond rules support the graphic modeling of decision-making scenarios. Decision trees proved to be very understandable for human beings. For modeling business situations, sense and respond rules use decision graphs which are an extension of decision trees for representing rules, thereby enhancing the understandability and expressiveness of rules, and shortening the learning curve for users.

Building Complex Rules with Divide and Conquer. Sense and respond rules can break down complex business situations in simple understandable conditions, which can be combined with each other for composing more complex conditions. The input of sense and respond rules are events and also the output are also so-called *response events*, which are raised when a rule fires. The fired events can be used as input for other (or even the same) rule for further evaluation, thereby effectively combining multiple rules.

Event Pattern Recognition. Event patterns are discovered when an event or multiple events occur that match the pattern's definition. Sense and respond rules allow to combine one or more event pattern with arbitrary event conditions in order to describe complex business situations.

Adaptability. Due to the graphical model and modular approach for constructing rules, sense and respond rules can be easily adapted to business changes. New event conditions or event patterns can be added or removed from the rule model in order to model changing business situations.

Service-oriented Rule Processing. Sense and respond rules are executed by event services, which supply the rule engine with events and process the evaluation result. Event services can run distributed on multiple machines and facilitate the integration with external systems.

3.1 Rule Characteristics

Sense and respond rules separate multiple aspects for the definition of rules which are specified and modeled separately. These three aspects are as follows:

Correlation. The definition of relationships and dependencies between events that are relevant for the rule processing are performed declaratively with correlation sets. With correlation sets, a rule engine is able to construct sequences of events that are applied to the condition defined in the rule. A business activity spanning some significant period of time can be represented by the interval between two or more events. For example, a transport might have a TransportStart and TransportEnd event-pair. Similarly, a shipment could be represented by the events ShipmentCreated, ShipmentDelivered and multiple transport event-pairs. A correlation set allows to associate such events by using the events' context data (e.g. ShipmentID, or TransportID).

Event/Condition/Action (ECA) Model. Sense and respond rules use a graphical model for describing constraints of events for business situations. ECA rules automatically perform actions in response to events provided stated conditions hold. The actions of sense and respond rules generate response events, which can be used triggering business activities or evaluating further rules.

Event Patterns. Event patterns complement the ECA model and allow separating the pattern matching logic from the action triggering. Event patterns solely describe business situations, such as event conditions or the discovery of missing events (e.g. a door is opened without closing it on time).

3.2 Rule Model

In the following, we present a sense and respond rule model for automatically detecting and responding to online betting fraud. For our example, we assume that a rule discovers a conspicuous number of high stake bets and checks betting limits. Scoring is used for measuring the overall number of high stake bets and betting limit overruns. When modeling the rule for the above-stated business problem, we first have to identify the events which have to be evaluated for the rule processing. For our example, we consider the following event types: *Bet Placed,* triggered when a customer places a bet on the betting platform, *Bet Won* signaling that a placed bet by a customer was won, and *Bet Lost* signaling that a placed bet by a customer was lost.

Figure 1 shows a sense and respond rule for the above-stated business scenario. Event condition shapes are used to check whether high stake bets occurred or betting limits have been reached. If an event condition evaluates to true, a customer score is increased by one. After increasing the score, it is checked against a threshold in another event condition shape. Based on the outcome of the threshold checking, betting alerts are generated. If there were too many high stake bets won and the betting limit was overrun multiple times, the customer account will be immediately blocked. On the right side of the figure, a correlation set (McGregor and Schiefer 2004) is shown which declaratively defines the relationships of the betting events. The rule engine needs the information of the correlation set in order to evaluate the event sequences of a particular player on the gambling platform.

The modeled sense and respond rule fully fulfills the stated business requirements. Nevertheless, there are some shortcomings in the first solution. The generation of alerts and the blocking of a customer account is directly linked to the event pattern matching logic. Another related shortcoming is that we had to model the high stake bet matching and betting limit overrun matching in one model in order to decide when to block a customer account. In the following, we show how to overcome these shortcomings with event pattern modeling.

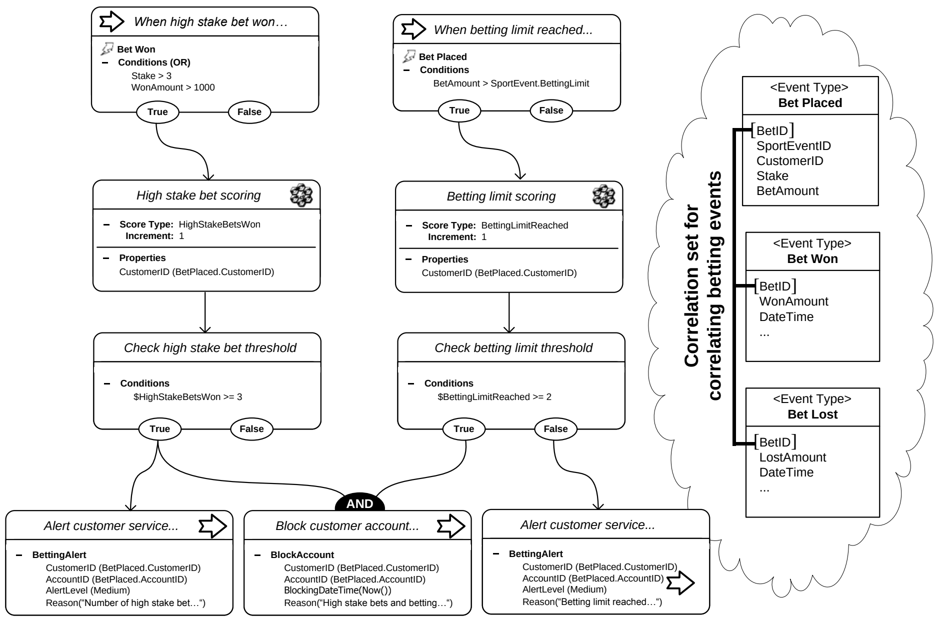
When high stake bet won…
Bet Won
Conditions (OR)
Stake > 3
WonAmount > 1000
True
False
When betting limit reached...
Bet Placed
Conditions
BetAmount > SportEvent.BettingLimit
True
False
High stake bet scoring
Score Type: HighStakeBetsWon
Increment: 1
Properties
CustomerID (BetPlaced.CustomerID)
Betting limit scoring
Score Type: BettingLimitReached
Increment: 1
Properties
CustomerID (BetPlaced.CustomerID)
Check high stake bet threshold
Conditions
$HighStakeBetsWon >= 3
True
False
Check betting limit threshold
Conditions
$BettingLimitReached >= 2
True
False
AND
Alert customer service...
BettingAlert
CustomerID (BetPlaced.CustomerID)
AccountID (BetPlaced.AccountID)
AlertLevel (Medium)
Reason("Number of high stake bet…")
Block customer account...
BlockAccount
CustomerID (BetPlaced.CustomerID)
AccountID (BetPlaced.AccountID)
BlockingDateTime(Now())
Reason("High stake bets and betting…")
Alert customer service...
BettingAlert
CustomerID (BetPlaced.CustomerID)
AccountID (BetPlaced.AccountID)
AlertLevel (Medium)
Reason("Betting limit reached…")
Correlation set for correlating betting events
<Event Type>
Bet Placed
BetID
SportEventID
CustomerID
Stake
BetAmount
<Event Type>
Bet Won
BetID
WonAmount
DateTime
...
<Event Type>
Bet Lost
BetID
LostAmount
DateTime
...

Figure 1: Sense and respond rule for reacting on betting fraud patterns

3.3 Event Pattern Modelling

In this section, we show how sense and respond rules can be broken down into reusable rule parts. In our previous example, we modeled two fraud scenarios in a single rule model. By defining each fraud scenario as a separate event pattern, it is possible to reduce the complexity of the rule and reuse the matching logic.

Figure 2 shows two event pattern models from the previous example. There is an event pattern model for high stake bet fraud and for betting limit fraud. Each of the fraud patterns has a pattern activator for collecting output information for the user of the pattern. A pattern model can have also input parameters. For instance, in the example, we define StakeLimit, WonAmountLimit and OverrunLimit as input parameters which can be set when using the pattern in a rule.

Next, we illustrate how the event patterns can be used in the rule model. Figure 3 shows a rule using the modeled event patterns. The event pattern shapes represent pattern proxies with input and output parameters. The input parameters can be set with a value or calculated by an expression. The output parameters can be used for bindings in event actions (e.g. attribute bindings of response events as shown in the figure). A pattern proxy is activated if the activator of the corresponding event pattern fires. As the figure shows, we have been able to encapsulate the fraud scenarios within two event patterns which can be parameterized and reused in a rule.

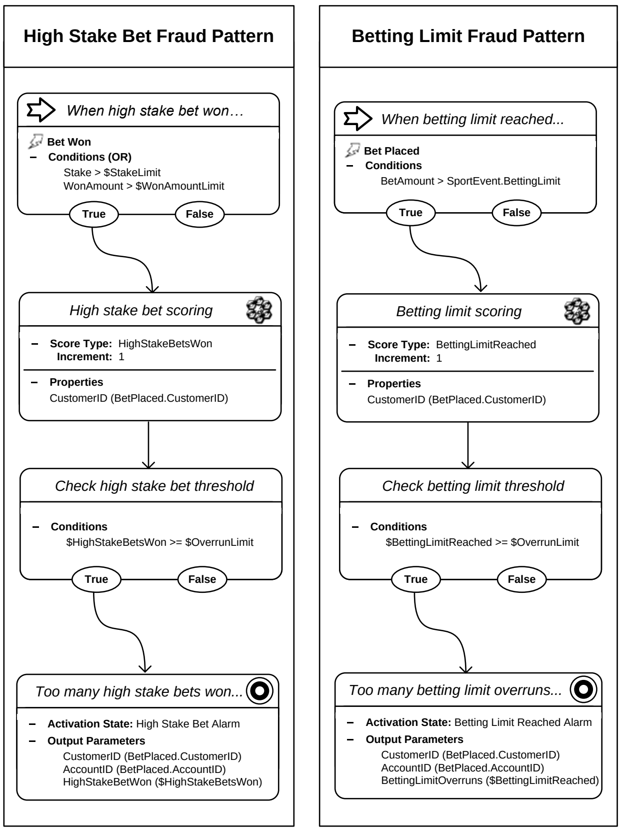
High Stake Bet Fraud Pattern
When high stake bet won…
Bet Won
Conditions (OR)
Stake > $StakeLimit
WonAmount > $WonAmountLimit
True
False
High stake bet scoring
Score Type: HighStakeBetsWon
Increment: 1
Properties
CustomerID (BetPlaced.CustomerID)
Check high stake bet threshold
Conditions
$HighStakeBetsWon >= $OverrunLimit
True
False
Too many high stake bets won...
Activation State: High Stake Bet Alarm
Output Parameters
CustomerID (BetPlaced.CustomerID)
AccountID (BetPlaced.AccountID)
HighStakeBetWon ($HighStakeBetsWon)
Betting Limit Fraud Pattern
When betting limit reached...
Bet Placed
Conditions
BetAmount > SportEvent.BettingLimit
True
False
Betting limit scoring
Score Type: BettingLimitReached
Increment: 1
Properties
CustomerID (BetPlaced.CustomerID)
Check betting limit threshold
Conditions
$BettingLimitReached >= $OverrunLimit
True
False
Too many betting limit overruns...
Activation State: Betting Limit Reached Alarm
Output Parameters
CustomerID (BetPlaced.CustomerID)
AccountID (BetPlaced.AccountID)
BettingLimitOverruns ($BettingLimitReached)

Figure 2: Separate event pattern models

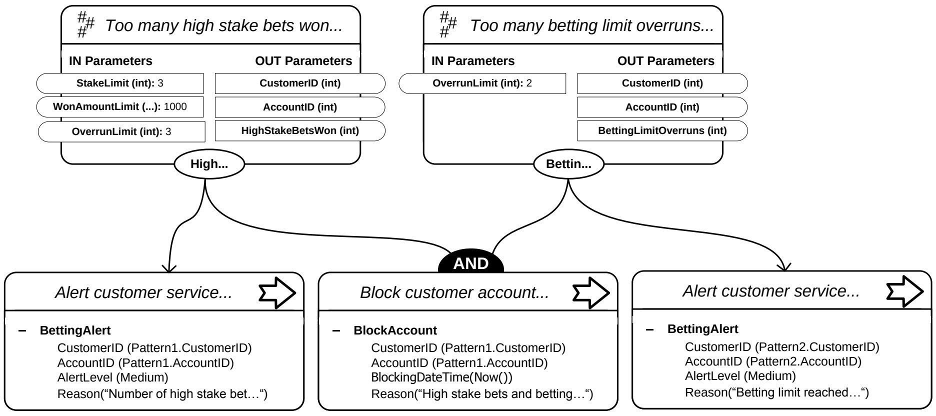
Too many high stake bets won...
IN Parameters
OUT Parameters
StakeLimit (int): 3
WonAmountLimit (...): 1000
OverrunLimit (int): 3
CustomerID (int)
AccountID (int)
HighStakeBetsWon (int)
High...
Too many betting limit overruns...
IN Parameters
OUT Parameters
OverrunLimit (int): 2
CustomerID (int)
AccountID (int)
BettingLimitOverruns (int)
Bettin...
AND
Alert customer service...
BettingAlert
CustomerID (Pattern1.CustomerID)
AccountID (Pattern1.AccountID)
AlertLevel (Medium)
Reason("Number of high stake bet…")
Block customer account...
BlockAccount
CustomerID (Pattern1.CustomerID)
AccountID (Pattern1.AccountID)
BlockingDateTime(Now())
Reason("High stake bets and betting…")
Alert customer service...
BettingAlert
CustomerID (Pattern2.CustomerID)
AccountID (Pattern2.AccountID)
AlertLevel (Medium)
Reason("Betting limit reached…")

Figure 3: Using event patterns in a rule

4 Discussion

4.1 Sense and Respond Rules

Sense and respond rules offer a new way for business users to define and manage the response to typical patterns of business events. A key advantage of sense and respond rules is that they allow to graphically model a comprehensive set of event conditions and event patterns without using nested or complex expressions. In contrast, SQL-based system for querying event streams require technical knowledge for specifying and changing event queries which makes them difficult to use for a wider range of users.

Sense and respond rules flexibly combine rule triggers which can be used to define IF-THEN-ELSE decisions. Event conditions and event cases can be checked for a true or false evaluation, and facilitates the implementation of "otherwise" situations. Complex decision scenarios are displayed with a decision graph which can delegate the decision-making to pattern models.

By describing correlation and event pattern aspects in a separate model, the definition of event conditions and patterns are simplified. Correlation sets capture the relationships between events and can also be defined with a graphical model. A separate event pattern model allows capturing the detection of business situations. The advantage of modeling and reusing event patterns are:

- Event patterns can be used parameterized and reused in various types of rules
- Rule complexity is reduced by breaking down large rule models into smaller rule parts
- Clean separation of event pattern matching logic and the binding logic of event actions.

These characteristics of modeling rules are essential when building rule sets for large industry solutions. Rule sets for industry solutions must be highly configurable and reusable for a wide range of business problems.

The service-oriented event processing system of SARI allows to flexibly link a rule service, processing sense and respond rules with other services. Event services are executed in parallel and controlled by the system. Services can be used to prepare the data for the rule processing as well as to process the response events generated by the rule service. The input and output for sense and respond rules are events which are delivered and processed by event services. Sense and respond rules, therefore, allow an easy integration within a service-oriented system environment.

However, the SARI system also has a drawback compared to SQL-based approaches for event stream processing. Using SQL for event stream queries allows to seamlessly integrate relational database systems by joining database tables with events that stream into the system. SARI requires services for preparing event data for the rule processing.

4.2 Event Processing with Rule Services

The SARI system uses an event processing model (EPM) for modeling event-driven processes. An EPM allows to integrate multiple services and adapters which can be used to implement event-driven processes. Dependent on the requirements and the business problem, the event services and adapters can be flexibly conjoined or disconnected. Links between the components and services represent a flow of events from one service to the next. The following issues are defined with the EPM:

- Structure for the processed events and data
- Configuration of services and adapters for processing steps, including their input and output parameters
- Interfaces to external systems for receiving data (Sense) and also for responding by executing business transactions (Respond)
- Data transformations, data analysis and persistence

A rule service is part of an EPM and can be configured with rule sets and linked with other services. EPMs allow to model which events should be processed by the rule service and how the response events should be forwarded to other event services. Figure 4 shows an EPM for the previous example. Data is collected and received from adapters which forward events to event services that consume them.

Initially the events are enriched in order to prepare the event data for the rule processing. A typical example would be the attachment of information about the customer (e.g. the source event only provides an account ID and the associated customer ID needs to be added). The rule service processes the enriched events

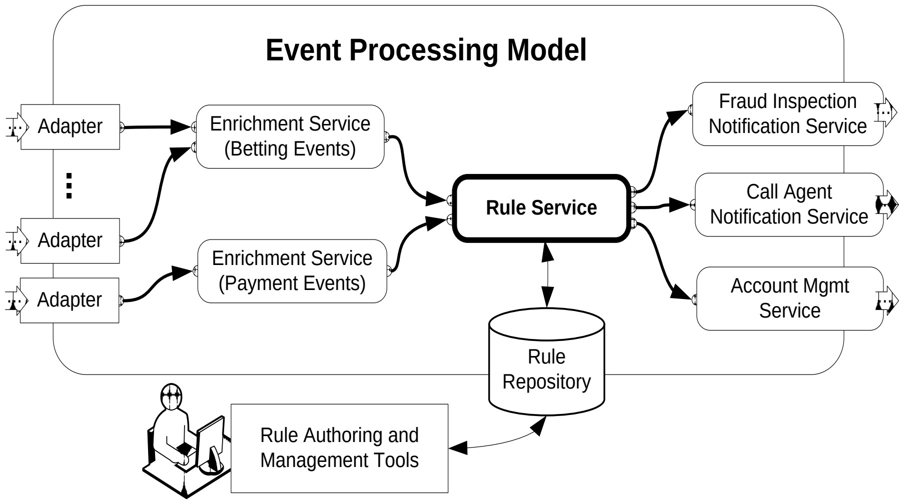
Event Processing Model
Adapter
Adapter
Adapter
Enrichment Service (Betting Events)
Enrichment Service (Payment Events)
Rule Service
Rule Repository
Fraud Inspection Notification Service
Call Agent Notification Service
Account Mgmt Service
Rule Authoring and Management Tools

Figure 4: Event processing model with rule service

according to the sense and respond rules and generates response events when a rule fires. The fired response events are published on the output port of the rule service and are forwarded to other services. As shown in Figure 4, the response events are sent to a service for sending notifications to fraud inspection department and call agents, or to services which allows to automatically changing the status of a user account. Sense and respond rules are stored and managed within a rule repository. SARI includes authoring and management tools which can be used by business users for graphically defining and modifying rule sets as introduced in the previous sections.

4.3 Rule Evaluation

During runtime, the rule service automatically correlates events emitted to the SARI system. Correlated event data is managed with correlation sessions (McGregor and Schiefer 2004), which are automatically activated before a triggering event is used for evaluating event conditions or patterns. Correlation sessions can be persistent and are used to maintain the current rule state. With correlation sessions, sense and respond rules can use the captured event data for accessing data of correlated events which has been previously processed. Furthermore, the sessions are used to maintain the state of activated ports of rule triggers.

The rule service uses a dependency graph of preconditions for determining whether an event action is executed. Figure 5 shows an example of a dependency graph with four rule triggers (circles) and one event action (rectangle). For this example, we assume that the rule triggers (1) and (2) are defined as OR precondition for rule trigger (3). As shown in the figure, initially the port of rule trigger (3) is activated, then the port of rule trigger (2) and (4).

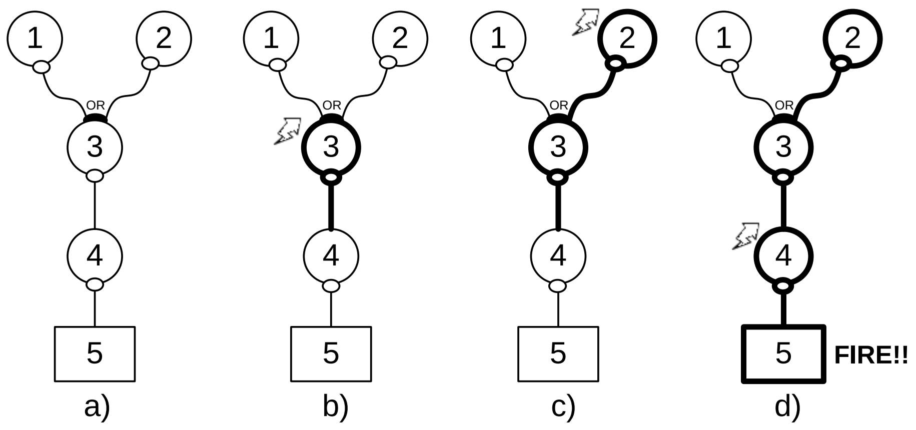

Figure 5: Dependency graph with activations of rule elements

As soon as all preconditions of an event action is fulfilled, the event action (5) fires. Please note, the rule service does not consider the order of rule trigger activations; for firing an event action is only relevant that the preconditions of its dependency graph evaluate to true.

5 Conclusion and Outlook

Event-based systems have been largely studied and used building and monitoring loosely coupled business solutions. This paper presented sense and respond rules for responding to business situations based on events which have been captured by an event-based system. Business situations are composed with event patterns and rule triggers which can be arbitrary combined and trigger actions when a rule fires. Sense and respond rules can be graphically modeled which makes it easier for business users to adapt rules for business changes. In runtime, the rules are processed by event services running on multiple machines and which can be seamlessly integrated in an event processing model.

The work presented in this paper is part of a larger, long-term research effort aiming at developing an event-driven rule management system. This system will allow users to model and manage comprehensive rule sets for industry solutions. A key focus of this future research work will be the visualization of events which have been processed with sense and respond rules.

References

Abadi DJ, Carney D, Cetintemel U, Cherniack M, Convey C, Lee S, Stonebraker M, Tatbul N, Zdonik S (2003) Aurora: A new model and architecture for data stream management. VLDB Journal 12, pp. 120–139.

Abadi DJ, Ahmad Y, Balazinska M, Çetintemel U, Cherniack M, Hwang JH, Lindner W, Maskey AS, Rasin A, Ryvkina E, Tatbul N, Xing Y, Zdonik S (2005) The Design of the Borealis Stream Processing Engine. In: Proc. of the Conf. on Innovative Data Systems Research, Asilomar, CA, USA, 277–289.

Bailey J, Crnogorac L, Ramamohanarao K, Sondergaard H (1997) Abstract Interpretation of Active Rules and its use in Termination Analysis, Proceedings of the Sixth International Conference on Database Theory (ICDT), Delphi, Greece.

Baralis E, Widom J (1994) An algebraic approach to rule analysis by means of triggering and activation graphs, In VLDB 94.

Barbará D, Mehrota S, Rusinkiewicz M (1994) INCAS: A Computation Model for Dynamic Workflows in Autonomous Distributed Environments. Technical Report, Department of Computer Science, University of Houston.

Bussler C, Jablonski S (1994) Implementing Agent Coordination for Workflow Management Systems Using Active Database Systems. Proc. 4th RIDE-ADS, Houston.

Chen SK., Jeng JJ, Chang H (2006) Complex Event Processing using Simple Rule-based Event Correlation Engines for Business Performance Management. CEC/EEE 2006, Palo Alto.

Clark T, Jones M, Armstrong C (2007) The Dynamic Structure of Management Support Systems. Theory, Development, Research Focus and Directions. MIS Quarterly Vol. 31 No. 3, 603- 607.

Davenport TH, Harris JG (2007) Competing on Analytics. The New Science of Winning, Boston.

Dayal U, Hsu M, Ladin R (1990) Organizing Long-Running Activities with Triggers and Transactions. Proc. SIGMOD, Atlantic City, NJ.

Eckerson W (2007), Best Practices in Operational BI. Converging Analytical and Operational Processes. TDWI Reserarch.

Esper (2007) http://esper.sourceforge.net, 2007-03-10.

Geppert A, Tombros D (1998) Event-Based Distributed Workflow Execution with EVE Proc. iFiP Int'l Conf. Distributed Systems Platforms and Open Distributed Processing (MIDDLEWARE '98).

Grigori D, Casati F, Castellanos M, Daysal U, Sayal M, Shan MC (2004) Business Process Intelligence. In: Computers in Industry, 53, S. 321-343.

Ingvaldsen JE, Gulla JA (2006) Model-Based Busines Process Mining. Information Systems Management, 23, S. 19-31.

Linders S (2008) Opportunities and limitations of using SOA concepts and technologies for building BI applications: a Delphi Study. University of Twende.

Luckham D (2005) The Power of Events, Addison Wesley.

Ludascher B (1998) Integration of Active and Deductive Database Rules, Phd thesis, University of Freiburg, Germany.

McGregor C, Schiefer J (2004) Correlating Events for Monitoring Business Processes, 6th International Conference on Enterprise Information Systems (ICEIS), Porto.

Sayal M, Casati F Dayal U, Shan MC (2002) Business Process Cockpit, Proceedings of the 28th VLDB Conference, HP.

Schiefer J, Seufert A (2005) Management and Controlling of Time-Sensitive Business Processes with Sense & Respond, International Conference on Computational Intelligence for Modelling Control and Automation, Vienna.

Seirio M, Berndtsson M (2005) Design and Implementation of an ECA Rule Markup Language. RuleML, Springer Verlag, pp. 98–112.

Van der Aalst W (2004) Business Alignment. Using Process Mining as a Tool for Delta Analysis. In: Proceedings of the 5th Workshop on Business Process Modelling, Development and Support (BPMDS), Riga.

Wu P, Bhatnagar R, Epshtein L, Bhandaru M, Shi Z (1998) Alarm correlation engine (ACE), In Proceedings of the IEEE/IFIP 1998 Network Operations and Management Symposium (NOMS), New Orleans.

Zdonik S, Stonebraker M, Cherniack M Cetintemel U, Balazinska M, Balakrishnan, H (2003) The Aurora and Medusa Projects. IEEE Data Engineering Bulletin, 26(1).