Correlating Business Events for Event-Triggered Rules

Josef Schiefer, Hannes Obweger, and Martin Suntinger

UC4 Senactive Software GmbH, Prinz-Eugen-Strasse 72,
1040 Vienna, Austria
{josef.schiefer,hannes.obweger,martin.suntinger}@uc4.com

Abstract. Event processing rules may be prescribed in many different ways, including by finite state machines, graphical methods, ECA (event-condition-action) rules or reactive rules that are triggered by event patterns. In this paper, we present a model for defining event relationships for event processing rules. We propose a so-called correlation set allowing users to graphically model the event correlation aspects of a rule. We illustrate our approach with the event-based system SARI (Sense and Respond Infrastructure) which uses correlation sets as part of rule definitions for the discovery of event patterns. We have fully implemented the proposed approach and compare it with alternative correlation approaches.

Keywords: Rule Management, Event Correlation, Event Processing.

1 Introduction

Operational business systems, such as enterprise resource planning and process management systems are able to report state changes within a business environment as business events. The state changes might occur with the execution of business operations or the completion of customer requests. These events can be used as independent triggers for activities as well as for reports or mining purposes using systems with an event-driven architecture (EDA). Event-based systems integrate a wide range of components into a loosely-coupled distributed system with event producers which can be application components, post-commit triggers in a database, sensors, or system monitors, and event consumers such as application components, device controllers, databases, or workflow queues. Thus, event-based systems are seeing increasingly widespread use in applications ranging from time-critical systems, system management and control, to e-commerce.

The process of defining a relationship between events and implementing actions to deal with the related events is known as event correlation. Event correlation in the business domain is a technique for collecting and isolating related data from various (potentially lower level) events in order to condense many events each holding a fraction of information into a meaningful business incident. The correlated information can then be used for discovering business patterns, such as business opportunities or exceptional situations. In many cases, this “aggregated” information can be published again as an event to the messaging middleware. Hence, series of enrichment steps are possible triggering activities on various levels of abstraction.

Many existing SQL-based approaches use joins for correlating events [1][3][5]. Thereby, events are processed as tuples which are joined when correlating them similar to the way it is done in relational databases. Although many people in the data management community are familiar with this approach, it has several drawbacks for event processing. Relational databases use foreign-key relationships and indices for creating optimal query execution plans. As such relationships are not available between event objects, it is difficult to build effective query execution plans for event streams. Within this paper, we show a correlation approach, which allows to model relationships between events for correlating them already during the event processing. Existing event-based systems do not allow the user to reuse correlations for defining rules in event-driven applications. With our approach, we try to fill the gap that existing event-based systems do not capture correlation information as separate meta-data and, thereby, loose many advantages for optimizing the rule processing. Event correlation is challenging due to the following reasons:

- Correlated business events can be processed by multiple rules for evaluating conditions, calculating metrics or discovering event patterns.
- Correlated events can occur at different points in time, requiring a temporary storage of correlated event data that is transparent for rules.
- Business events occur in various source systems and different formats. Nonetheless, they must be captured and correlated with minimal latency and minimal operational system impact.
- The event correlation should be independent from the execution systems or protocols.
- Late arrival of events due to network failures or downtimes of operational systems within heterogeneous and distributed software environments are common situations that have to be considered for the event correlation and for the rule processing.
- Only relevant event data for applications and users must be unified, transformed, and cleansed before they are correlated. In many situations, only a small set of selected event attributes are needed for the correlation.

The presented approach is generic and thus applicable in various application domains. Throughout the paper we discuss different application examples based on real-world business scenarios.

The remainder of this paper is organized as follows: Section 2 discusses related work. In Section 3, we discuss event-triggered rules and show how they are used in event-based system. Section 4 discusses correlation sets and correlation bands for modeling event relationships. How correlation sets and correlation bands are managed in runtime is discussed in Section 5. We furthermore present a real-world application from the logistics domain in Section 6. Finally, in Section 7, we conclude this paper and provide an outlook to future research.

2 Related Work and Contribution

The key characteristic of an event-based system is its capability of handling complex event situations, detecting patterns, aggregating events and making use of time windows for collecting event data over a period of time. Event correlation plays a crucial role for performing these tasks. In the following, we provide an overview of the correlation capabilities of event processing engines.

Esper [5] is an Open Source event stream processing solution for analyzing event streams. It has a lightweight processing engine and is currently available under GPL licence. Esper supports conditional triggers on event patterns and SQL queries for event streams. Event correlation can be performed by joining the attributes of events.

Borealis and Aurora [1] are further examples of stream processing engines for SQL-based queries over streaming data with efficient scheduling and QoS delivery mechanisms. Medusa [17] focuses on extending Aurora's stream processing engine to distribute the event processing. Borealis extends Aurora's event stream processing engines with dynamic query modification and revision capabilities and makes use of Medusa's distributed extensions. All of these stream engines use joins for correlating events.

RuleCore [12][14] is an event-driven rule processing engine supporting Event Condition Action (ECA) rules and providing a user interface for rule building and composite event definitions. In RuleCore, event correlation settings are embedded in the rule definition, which are used by the rule engine during the event processing. Chen et al. [4] show an approach for rule-based event correlation. In their approach, they correlate and adapt 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.

AMIT [2] is an event stream engine whose goal is to provide high-performance situation detection mechanisms. AMIT offers a sophisticated user interface for modelling business situations based on the following four types of entities: events, situations, lifespans and keys. In AMIT, lifespans allow the definition of time intervals wherein specific patterns of correlated events can be detected.

Detecting and handling exceptional events also plays a central role in network management [6]. Alarms indicate exceptional states or behaviors, for example, component failures, congestion, errors, or intrusion attempts. Often, a single problem will be manifested through a large number of alarms. These alarms must be correlated to pinpoint their causes so that problems can be addressed effectively. Many existing approaches for correlating events have been developed from network management. Event correlation tools help to condense many events, which individually hold little information, to a few meaningful composite events.

Rule-based analysis is a traditional approach to event correlation 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 [16].

Another group of approaches incorporate an explicit representation of the structure and function of the system being diagnosed, providing information about dependencies of components in the network [8] or about cause-effect relationships between network events. The fault discovery process explores the network model to verify correlations between events. NetFACT [7] uses an object-oriented model to describe the connectivity, dependency and containment relationships among network elements. Events are correlated based on these relationships. Nygate [11] models the cause-effect relationships among events with correlation tree skeletons that are used for the correlation.

In spite of intensive research in the past years for correlating events in the event and network management domain, there is, to our knowledge, no existing approach that allows to model relationships between business events for correlation purposes in order to support functions for the event processing. In many event-based systems, the relationship information of events is buried in query statements or programming code.

This paper is an attempt to show an approach which allows to externalize event relationship information in order to utilize this information for later reuse. We believe that information on how events are correlated is fundamental for processing and analyzing events and is, in current event-based systems, not appropriately managed. We propose a correlation model for defining relationships between events which can be used by event-based system for rule processing tasks which require correlated events. With our approach, event-based systems are able to capture correlation concerns with a separate model without increasing the complexity of the rule model. The captured correlation information is then used the processing of various rules.

3 Event-Triggered Rules

Event-triggered rules prescribe actions to be taken whenever an instance of a given event pattern is detected. While the typical users of event stream queries are developers, event-triggered rules try to describe event patterns on a more abstract level and link event-patterns with business actions. Event-triggered rules require event correlation for the detection of event patterns over a time period as well as for tracing the actions triggered by rules [13].

In an event-based system, information about business activities is encapsulated in events, which capture attributes about the context when the event occurred. Event attributes are items such as the agents, resources, and data associated with an event. For example, in an online betting scenario, a typical bet placement event could have the following attributes as context information: account ID, amount, market, league, sport event ID, odds, and bet type. For the rule processing, it is important to correlate temporally and semantically related events. Event attributes, i.e., data from the context of an event, can be used to define such relationships.

The rule engine uses the correlated events for discovering event patterns. When event patterns are matched, the rule engine instantly responds to the source systems. Fig. 1 illustrates the system architecture and essential data processing flows from the source system to the event-based system and vice-versa. The event-based system receives events on notable occurrences such as bet placements via unified interfaces to the betting platform and legacy systems. It can evaluate the received events and, if necessary, respond in real time by carrying out automated decisions [6]. The communication between the source systems and the event-based system is always asynchronous, avoiding a tight coupling between the systems.

Transactional data, such as a bet placement, is propagated and continuously integrated as an event stream. The event-based system correlates the events of the event stream and continuously calculates metrics and scores. Finally, the rule engine of event-based systems applies ECA rules on the correlated event sequences and calculated metrics and scores.

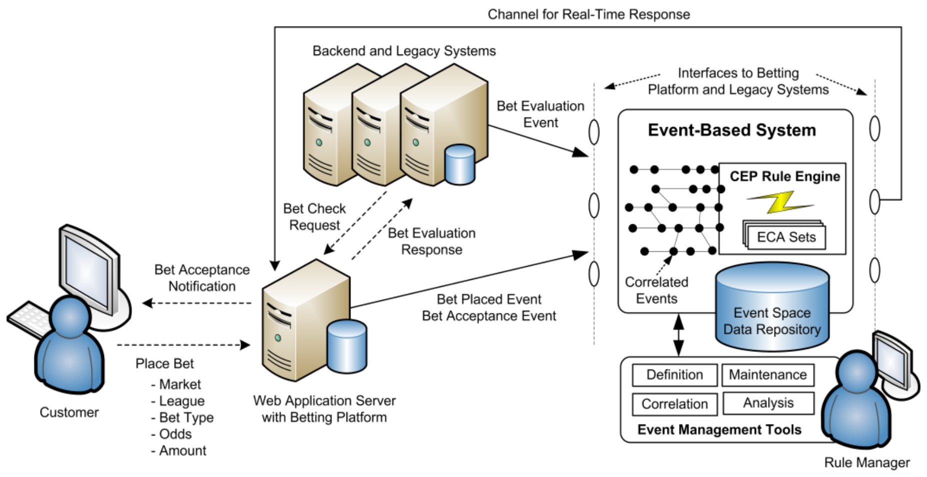

Fig. 1. Event-Driven Rule Engine for Monitoring a Betting Platform

In the following, we show a model for defining event relationships for event-triggered rules with so-called *correlation sets*. The rules use correlation sets for defining correlation concerns in separate model. During runtime the rule engine uses the rule and correlation model for the pattern detection.

4 Correlation Sets

An activity spanning a period of time is 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, ShipmentShipped and multiple transport event-pairs. For purposes of maintaining information about business activities, events capture attributes about the context when the event occurred. Event attributes are items such as the agents, resources, and data associated with an event, the tangible result of an action (e.g., the result of a transport decision), or any other information that gives character to the specific occurrence of that type of event. Elements of an event context can be used to define a relationship with other events. We use *correlation sets* for modeling these event relationships between events of business activities.

In SARI, correlations between events are declaratively defined in a correlation model (correlation sets), which is used by the correlation engine during runtime. Correlation sets are able to define relationships based on matching methods for correlating attributes among event types. Correlation sets have a set of *correlation bands* which define a sequence of events which use the same matching approach for the event correlation. For instance, if we want to correlate order events with transportation events (of the same order), we might have one band defining the correlation between order events (e.g. OrderCreated, OrderShipped, OrderFulfilled) and a second band defining the correlation between transport events (e.g. TransportStart, TransportEnd).

SARI supports various types of correlation bands, such as elementary correlation, knowledge-based correlation, self-referencing correlation, language-specific correlation

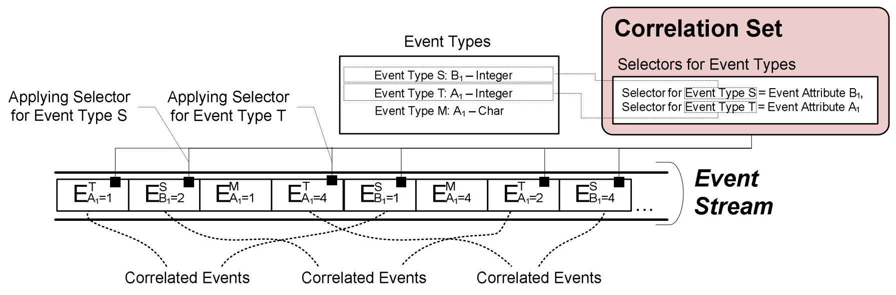
Correlation Set
Selectors for Event Types
Selector for Event Type S = Event Attribute B_1,
Selector for Event Type T = Event Attribute A_1
Event Types
Event Type S: B_1 – Integer
Event Type T: A_1 – Integer
Event Type M: A_1 – Char
Applying Selector for Event Type S
Applying Selector for Event Type T
$E^T_{A_1=1}$
$E^S_{B_1=2}$
$E^M_{A_1=1}$
$E^T_{A_1=4}$
$E^S_{B_1=1}$
$E^M_{A_1=4}$
$E^T_{A_1=2}$
$E^S_{B_1=4}$
…
Event Stream
Correlated Events
Correlated Events
Correlated Events

Fig. 2. Illustrates the application of a method of correlating events in a stream of events where E represents an event. Each event E has an identifier T, S, M etc. for the type of its event and at least one attribute A_1, B_1, etc. of this event.

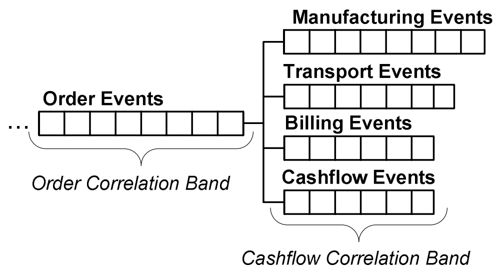

Fig. 3. Shows a correlation set with multiple correlation bands correlating events from different sources or domains. The idea of correlation bands is to capture semantically closely related events in a separate correlation arrangement. The correlation set is able to link multiple bands, thereby correlating the events of *all* its bands.

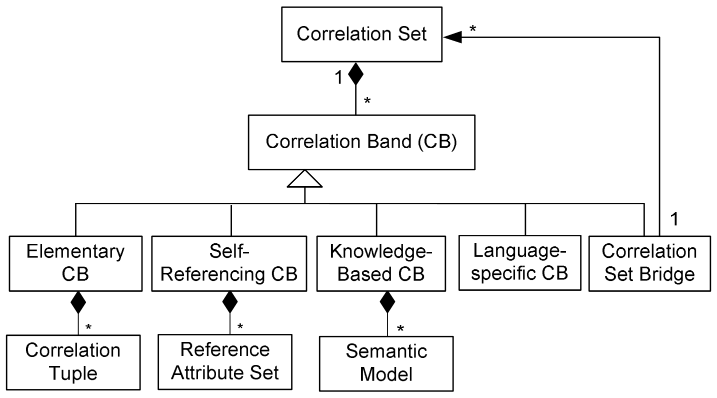

Fig. 4. Correlation Meta Model

and bridges for correlation sets. Fig. 4 shows the meta model for correlation sets. An elementary correlation band (CB) uses correlation tuples for defining relationships between event types based on matching event attributes. Knowledge-based correlation bands use a semantic model for defining the event relationships. Self-referencing correlation bands associate events based on reference information held by events (in other words, each event "knows" about its correlated events). Language-specific correlation bands use programming or query languages for defining event relationships. Correlation set bridges are a special kind of correlation bands which function as proxies for chaining multiple correlation sets. In other words, a correlation set bridge establishes a link between stand-alone correlation sets for correlating them. In the following sections, we define the various types of correlation bands in more detail.

4.1 Elementary Correlation Band

Elementary correlation bands define a direct relationship between events based upon matching event attributes. An event stream consists of events which conform to event types $T = \{t_1, t_2, \dots, t_n\}$. Each event type t_i in T defines a set of event attributes A_i. A relationship between events is defined by associating an attribute $a_j \in A_j$ of an event type t_j with attributes of one or more other events. Formally, an elementary correlation set is defined as follows:

Elementary Correlation Band $P = (T, \mathcal{A}, \mathcal{C})$ where:

- T: a set $\{t_1, t_2, \dots, t_n\}$ of event types of an event stream.
- $\mathcal{A}$: a set $\{A_1, A_2, \dots, A_n\}$ of sets of attributes as defined by the event types in T. Each event type has its own set of attributes.
- $\mathcal{C}$: a set $\{C_1, C_2, \dots, C_m\}$ of correlation tuples $C_i = \left(a_{f_k}, a_{g_l}, \dots\right)$, $a_{i_j} \in A_i$, which define relationships between one or more event types from T by associating their attributes.

Fig. 5 shows a relationship between TransportStart and TransportEnd events. The relationship is defined by associated attributes (TransportId) of the two event types. A relationship may include one or more correlating attributes which are part of correlation tuples. In other words, a correlation tuple defines attributes from different event types which have to match in order to correlate. A correlation set may include one or more event types which define relationships with correlation tuples.

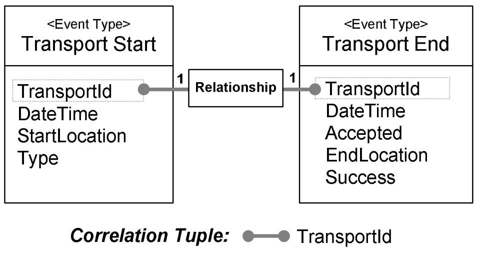

Fig. 5. Elementary Correlation Band Example

4.2 Self-referencing Correlation Band

When using self-referencing correlation bands, relationships between events are defined by the correlated events themselves. In self-referencing correlations, each event is able to create a link to other events by holding a unique identifier of each related event. For elementary correlation bands, we used attributes of the event context for modeling event relationships. Thereby, a user has to know which event attributes are appropriate for defining the relationships. Self-referencing correlation bands do not make this assumption and allow events to directly reference other events. Formally, a self-referencing correlation set is defined as follows:

Self-Referencing Correlation Band $S = (T, \mathcal{A}, a, R)$ where:

T: a set of event types of an event stream.

$\mathcal{A}$: a set $\{A_1, A_2, \ldots, A_n\}$ of sets of attributes as defined by the event types in T. Each event type has its own set of attributes.

a: an attribute that uniquely identifies an event. This attribute must be shared among all event types, i.e., $a \in A_i \; \forall \; i = 1, \ldots, n$.

R: a set of attributes $\{r_1, r_2, \ldots, r_m\}$, $r_i \in \bigcup_{j=1}^{n} A_i$, that is used to define relationships between events by having the value of another event's a-attribute.

In the following example, we extend our previous elementary correlation band of transport events with the corresponding shipment events ShipmentCreated and ShipmentShipped. For the correlation of the two shipment events, each ShipmentCreated event directly references a ShipmentShipped event using a unique event identifier.

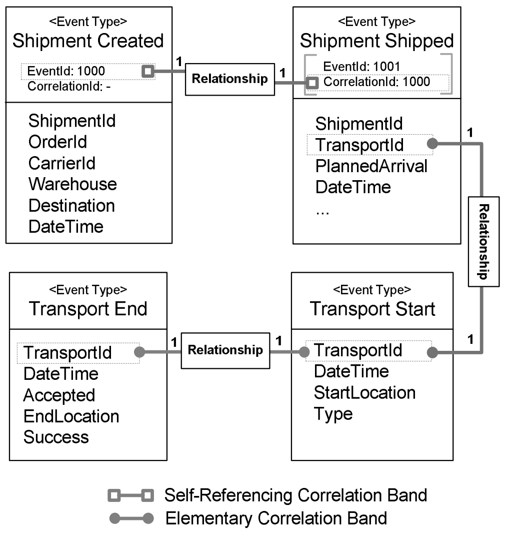

Fig. 6. shows a self-referencing correlation band. It includes event types with additional header attributes for a global unique event ID and a correlation ID which is used to directly reference other events. The advantage of self-referencing correlation sets is that relationships between events can be defined independently of the event context (which is given by the event attributes). Nevertheless, a consequence from wiring events directly is that the event producer has to track any generated events in order to be able to set the correct references for the event objects.

4.3 Knowledge-Based Correlation Band

A knowledge-based correlation set is a special case of an elementary correlation set. It defines an inferred relationship between events and associates a set of events by using semantic knowledge for correlating them. In other words, instead of directly associating a set of events by defining their relationships based solely upon the event attributes, knowledge-based correlation sets use algorithms which allow incorporating external knowledge for correlating events. These algorithms can use external databases, semantic networks or knowledge maps for inferring a relationship between events. Formally, a knowledge-based correlation set is defined as follows:

Knowledge-Based Correlation Band $K = (T, \mathcal{A}, k)$ where:

T: a set of event types of an event stream.

$\mathcal{A}$: a set $\{A_1, A_2, \ldots, A_n\}$ of sets of attributes as defined by the event types in T. Each event type has its own set of attributes.

k: a knowledge base which uses sets of attributes from $\mathcal{A}$ to define relationships between events by inferring that two or more events are correlating.

The following example (see Fig. 7) shows two event types with an inferred relationship. We assume that we want to correlate the events for job openings and job applications which have a matching job description and objectives. Since the job description of a company and the job objectives of an application are usually captured as full-text, an algorithm has to decide based upon a semantic model whether a job description and the job objectives of an application are matching.

An approach for semantic event correlation was presented by Moser et al. [10]. They present three types of semantic correlation: 1) basic semantic correlation, 2) inherited semantic correlation, and 3) relation-based semantic correlation. For the first two kinds of semantic correlations, ontologies are used to find matching terms that share the same (inherited) meaning. Relation-based semantic correlations use modeled relations defined in ontologies for the correlation process.

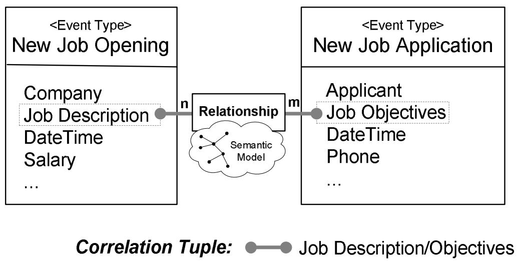

Fig. 7. Knowledge-Based Correlation Band

4.4 Language-Specific Correlation Band

Language-Based correlation bands use a language for defining the correlation. Many existing event-based systems use this type of correlation for querying event streams [1][5]. SQL-based query languages have become popular in recent years, which use join operations in order achieve event correlations. When using language-based correlation bands, only the correlation capabilities of the language are being used. In other words, language-based correlation bands use a language to specify relationships between events. The relationship information is embedded within the language and only the correlation or query engine is able to interpret the language to perform the correlation.

When using SQL-based query languages, the correlation band would use join operations for defining relationships between events. The language must be able to accept a set of input event streams and must generate an output stream with the correlated event data. The following example shows the correlation of customer and supplier events which occur within a certain time window [5].

```
SELECT A.transactionId, B.customerId,
       A.supplierId,
       B.timestamp - A.timestamp,
FROM   TxnEventA.win:time(30 minutes) A,
       TxnEventB.win:time(30 minutes) B
WHERE  A.transactionId = B.transactionId
```

An advantage of a language-based approach is that the language can be used for manipulations, calculations, or filtering of event data which is required for the correlation process. In addition, many SQL-based languages gain further expressiveness by providing powerful time window or join operations.

4.5 Correlation Set Bridge

A correlation set bridge allows linking existing correlation sets, each having its own set of correlation bands. Thereby, correlation set bridges allow combining existing correlation sets in order to extend their correlation scope. Formally, a bridged correlation set is defined as follows:

Correlation Set Bridge $B = (\mathcal{S})$ where:
$\mathcal{S}$: a set of correlation sets which should be linked.

The correlation set bridge will create a super-correlation set which includes all correlation bands of its child correlation sets. We want to extend the previous example with self-referencing correlation bands. We assume that the self-referencing correlation band of shipment events and all transport related events have been modeled in a separate correlation set. By using correlation set bridges, shipment events can be correlated with the transport events by virtually creating a super correlation set including all correlation bands. Conjunct event types (in our example the Shipment-Shipped event type) are used to create the bridge and expand the correlation scope by reusing existing correlation sets.

5 Correlating Events with Correlation Sets

In the following, we present an overview of the SARI correlation engine for correlating events with correlation sets in a distributed computing environment. Event correlation requires state information about the events being processed in order to determine which events are correlating. For each correlation band of a correlation set, SARI is creating a correlation session for collecting state information. If the correlation engine is able to link correlation bands, the corresponding sessions will be merged. In other words, correlation state information can evolve over time for different correlation bands which are combined as soon as events are correlated.

Another advantage of using correlation sessions is that they enable event correlation within a distributed and continuous event stream processing environment. Fig. 8 shows a distributed event-based system with a correlation service that manages correlation sessions for multiple nodes. Each node of the system performs event processing tasks which might require event correlations. When an event correlation is necessary, the node will access the correlation service in order to activate a session for each set of correlating events. The correlation service will synchronize the session access. The correlation session can be used by the node to store arbitrary data such as the correlated events or only some of their key information. For further details on managing the lifecycle of correlation sessions for distributed event-based systems refer to McGregor et. al [9].

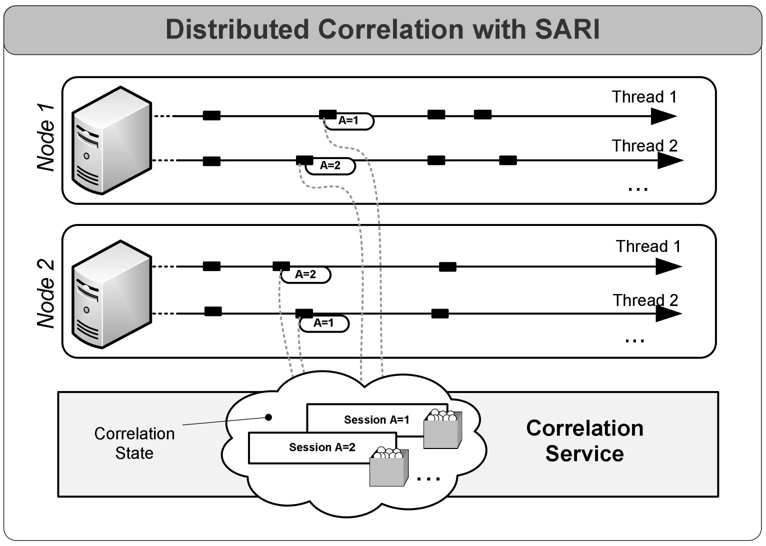

Fig. 8. Managing Correlations with Sessions

6 SARI Application Scenario

In this section, we present an example from the transportation and logistics domain. We show the modeling of SARI rules as well as how to validate and visualize rule processing results.

Fig. 9 shows the example's correlation set in the graphical rule editor of SARI. With the SARI rule editor, users can easy link attributes from a set of event types. Thereby, the rule editor automatically generates the resulting correlation band. Below the graphical editor, the resulting correlation bands are listed. For this example, we have separate correlation bands for both the transport and the shipment, which are combined by the correlation set. The modeled correlation set can be now assigned to an arbitrary number of SARI rules for detecting patterns from a correlated set of events. SARI rules allow defining complex event patterns, which use event conditions and timers for triggering response actions. Event conditions and timers can be arbitrarily combined with logical operators in order to model complex situations. Response actions are triggered when preconditions evaluate to true.

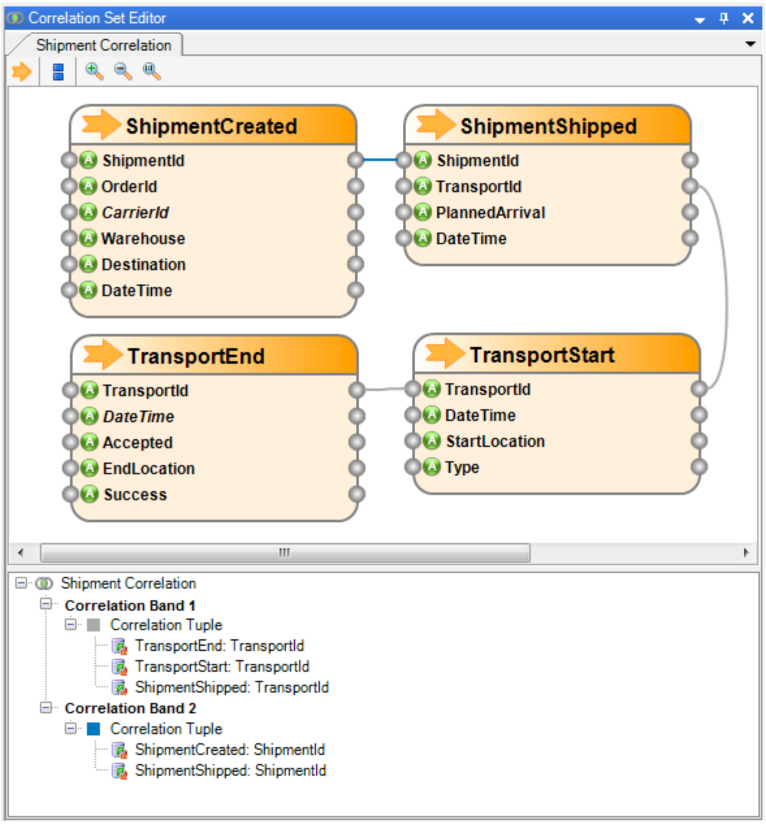

Fig. 9. Correlation Set Editor

Fig. 10 shows a SARI rule which monitors the transports of carrier 1200 airfare transports from Munich. If the shipment arrives more than 30 minutes late, two response actions are triggered: 1) an alert event is generated, and 2) a carrier satisfaction score is decreased. For correctly correlating the shipment and transport events, a user can refer to the correlation set from the previous section in the rule properties. A detailed discussion on the elements of SARI rules is given by Schiefer et. al in [13].

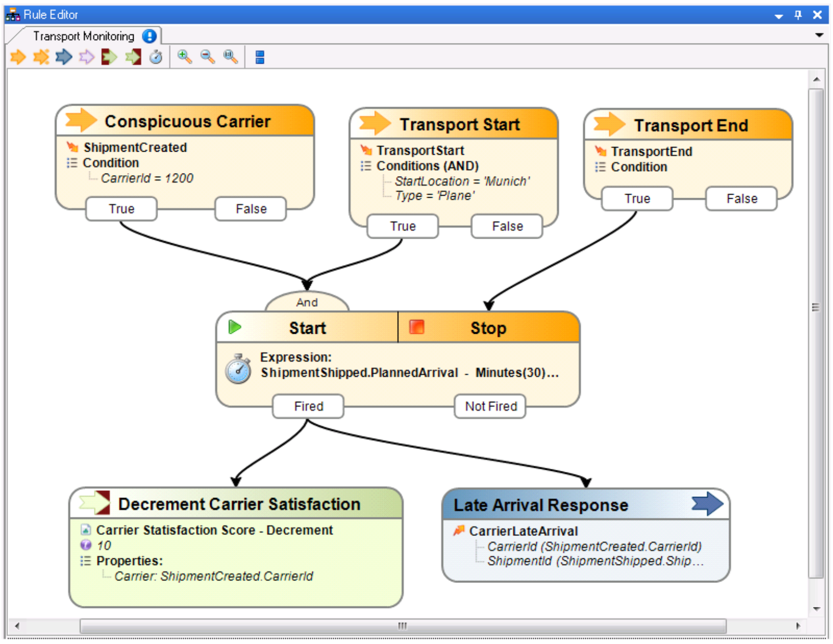

Fig. 10. SARI Rule – Transport Monitoring

6.1 Visualization of Rule Processing Results

SARI provides interactive visualizations of event streams to support business analysts in exploring business incidents. In the following, we show the event-tunnel visualization framework [15] for visualizing correlated events and validating rules. The event-tunnel is based on the metaphor of considering the event stream as a cylindrical tunnel, which is presented to the user from multiple perspectives. The visualization is able to display relationships between events which can be used for discovering root causes and causal dependencies of event patterns. The event-tunnel supports different types of placement policies for arranging events in the event tunnel. The centric event-sequence placement-policy (CESP policy), for instance, is focused on sequences of correlated events, such as the events of a business process instance. The policy plots event sequences over time (from the tunnel inside to the outer rings) and avoids overlapping events by clock-wise positional shifts. This technique results in characteristic patterns being abstract visual representations of business transactions.

Fig. 12 shows a screenshot of the 3D event-tunnel view for the events of a shipment process. While bullets represent events, the colored bands between them highlight correlation bands. The display of correlated events enables an abstract representation of underlying business transactions with characteristic patterns. If a rule generated some output (for instance an alert), it is displayed as part of the correlated event sequence. Business analysts can highlight events with characteristics of interest (e.g., delayed shipments, shipments with a triggered alert) for deeper analysis. Based on event correlations, it is possible to drill through the data, navigating to related

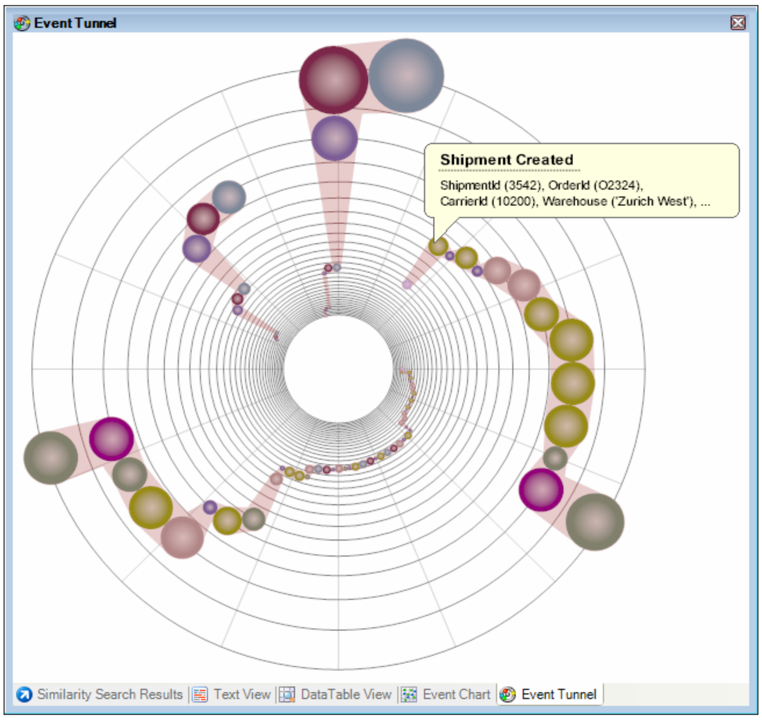
Event Tunnel
Shipment Created
ShipmentId (3542), OrderId (O2324),
CarrierId (10200), Warehouse ('Zurich West'), ...
Similarity Search Results
Text View
DataTable View
Event Chart
Event Tunnel

Fig. 11. Event-Tunnel Visualization of Correlated Events

(i.e., correlated) events which enables intuitive tracking of business incidents and root cause analyses. For more information on the event-tunnel, the CESP algorithm and evolving graphical patterns, readers may refer to Suntinger et. al [15].

7 Conclusion and Future Work

In large organizations, huge amounts of data are generated and consumed by business processes or human beings. Business managers need to respond with up-to-date information to make timely and sound business decisions. This paper described an approach for modeling relationships between events with correlation sets with the aim of correlating data from event streams for rule processing. We introduced various types of correlation sets and illustrated the usage of correlation sets with the SARI system and various types of applications. Our approach eases the separation of correlation concerns from rules, thereby making them more manageable and useful in a wide range of contexts. The work presented in this paper is part of a larger, long-term research effort aiming at developing an event stream management platform called SARI. A key focus of this future research work will be the automatic discovery of relationships and similarities between events and event sequences.

References

1. Abadi, D.J., Ahmad, Y., Balazinska, M., Çetintemel, U., Cherniack, M., Hwang, J.H., Lindner, W., Maskey, A.S., Rasin, A., Ryvkina, E., Tatbul, N., Xing, Y., Zdonik, S.: The Design of the Borealis Stream Processing Engine. In: Proc. of the Conf. on Innovative Data Systems Research, Asilomar, CA, USA, pp. 277–289 (2005)
2. Adi, A., Etzion, O.: AMIT - the situation manager. The VLDB Journal 13(2), 177–203 (2004)
3. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and Issues in Data Stream Systems. In: Proc. of 21st PODS, Madison, Wisconsin (May 2002)
4. Chen, S.K., Jeng, J.J., Chang, H.: Complex Event Processing using Simple Rule-based Event Correlation Engines for Business Performance Management. In: CEC/EEE (2006)
5. Esper, `http://esper.sourceforge.net` (2007-03-10)
6. Feldkuhn, L., Erickson, J.: Event Management as a Common Functional Area of Open Systems Management. In: Proc. IFIP Symposium on Integrated Network Management. North-Holland, Amsterdam (1989)
7. Houck, K., Calo, S., Finkel, A.: Towards a practical alarm correlation system. In: IEEE/IFIP Symposium on Integrated Network Management (1995)
8. Katzela, I., Schwartz, M.: Schemes for fault identification in communication networks. IEEE Transactions on Networking (1995)
9. McGregor, C., Schiefer, J.: Correlating Events for Monitoring Business Processes. In: 6th International Conference on Enterprise Information Systems (ICEIS), Porto (2004)
10. Moser, T., Roth, H., Rozsnyai, S., Biffl, S.: Semantic Event Correlation Using Ontologies. In: 3rd Central and East European Conference on Software Engineering Techniques (CEE-SET 2008), Brno (2008)
11. Nygate, Y.A.: Event correlation using rule and object base techniques. In: IEEE/IFIP Symposium on Integrated Network Management (1995)
12. RuleCore, `http://www.rulecore.com/` (2007-03-10)
13. Schiefer, J., Szabolcs, R., Saurer, G., Rauscher, C.: Event-Driven Rules for Sensing and Responding to Business Situations. In: International Conference on Distributed Event-Based Systems, Toronto (2007)
14. Seirio, M., Berndtsson, M.: Design and Implementation of an ECA Rule Markup Language. In: RuleML. Springer, Heidelberg (2005)
15. Suntinger, M., Obweger, H., Schiefer, J., Gröller, E.: The Event Tunnel: Exploring Event-Driven Business Processes. IEEE Computer Graphics and Applications (October 2008)
16. Wu, P., Bhatnagar, R., Epshtein, L., Bhandaru, M., Shi, Z.: Alarm correlation engine (ACE). In: Proceedings of the IEEE/IFIP 1998 Network Operations and Management Symposium (NOMS), New Orleans (1998)
17. Zdonik, S., Stonebraker, M., Cherniack, M., Cetintemel, U., Balazinska, M., Balakrishnan, H.: The Aurora and Medusa Projects. IEEE Data Engineering Bulletin (2003)