Beyond process monitoring: a proof-of-concept of event-driven business activity management

by Christian Janiesch, Martin Matzner, and Oliver Müller

Fact sheet

Title	"Beyond process monitoring: A proof-of-concept of event-driven business activity management"
Authors	Christian Janiesch, Martin Matzner, and Oliver Müller
Year	2012
Status	Published
Full citation	Janiesch, C., Matzner, M., & Müller, O. (2012). Beyond process monitoring: a proof-of-concept of event-driven business activity management. *Business Process Management Journal*, *18*(4), 625–643. doi:10.1108/14637151211253765

Structured Abstract:

Purpose. The purpose of this paper is to show how to employ complex event processing (CEP) for the observation and management of business processes. It proposes a conceptual architecture of BPM event producer, processor, and consumer and describes technical implications for the application with standard software in a perfect order scenario.

Design/methodology/approach. The authors discuss business process analytics as the technological background. The capabilities of CEP in a BPM context are outlined an architecture design is proposed. A sophisticated proof-of-concept demonstrates its applicability.

Findings. The results overcome the separation and data latency issues of process controlling, monitoring, and simulation. Distinct analyses of past, present, and future blur into a holistic real-time approach. The authors highlight the necessity for configurable event producer in BPM engines, process event support in CEP engines, a common process event format, connectors to visualizers, notifiers and return channels to the BPM engine.

Research limitations. Further research will thoroughly evaluate the approach in a variety of business settings. New concepts and standards for the architecture's building blocks will be needed to improve maintainability and operability.

Practical implications. Managers learn how CEP can yield insights into business processes' operations. The paper illustrates a path to overcome inflexibility, latency, and missing feedback mechanisms of current process modeling and control solutions. Software vendors might be interested in the conceptualization and the described needs for further development.

Originality/value. So far, there is no commercial CEP-based BPM solution which facilitates a round trip from insight to action as outlines. As major software vendors have begun developing solutions (BPM/BPA solutions), this paper will stimulate a debate between research and practice on suitable design and technology.

Keywords: Business process management, Business process re-engineering, Computer software, Complex event processing, Business activity monitoring, Architecture, Standard software

Paper type: technical paper

Beyond process monitoring: a proof-of-concept of event-driven business activity management

1 Introduction

Business processes represent the core of an enterprise's value creation. Thus, a thorough management of business process performance makes a —if not the— major contribution to business success (Hammer & Champy, 1993).

The application of various measurement and analysis techniques on process-related data has been proposed under umbrella terms such as process intelligence, process mining, or process analytics. The aim of these approaches is to provide process owners and participants with evidence-based insights about the performance of operational business processes (Davenport & Harris, 2007; zur Muehlen & Shapiro, 2010). Business Process Management System (BPMS) constitute a primary data source for these approaches. To meet legal regulations, BPMS have to provide support for logging processed transactions. Consequently, these systems provide some kind of log to track the execution of processes. The data contained in these logs represents a rich source of information to gain insight into executed and running processes and to take proactive or corrective actions to improve process performance.

However, a review of today's process-aware enterprise systems shows: Many BPMS are lacking sophisticated capabilities to analyze log data (Cheung & Hidders, 2011). Moreover, Business Activity Monitoring (BAM) or Process Mining tools (van der Aalst, 2011; Weske, van der Aalst, & Verbeek, 2004; zur Muehlen & Shapiro, 2010) are limited to rather passive

monitoring and reporting and are not able to actually take action on a running business process. It would require hardwiring systems of these two kinds to achieve end-to-end insight to action feedback loops. Evidently, an architecture built on hardwired components inescapably suffers from inflexibility, for instance, in use cases with multiple, distributed process-aware enterprise systems.

Against this background, we have investigated the feasibility of applying the concept of Complex Event Processing (CEP) to BPMS in order to allow for closed-loop monitoring and control of business processes. CEP, in general, comprises a set of techniques for making sense of the behavior of a monitored system by deriving higher-level knowledge from lower-level system events in a timely and online fashion (Eckert, 2008; Etzion & Niblett, 2010; Luckham, 2002). Following the methodology for design science proposed by Peffers, Tuunanen, Rothenberger, and Chatterjee (2007), we developed a conceptual architecture based on prior literature on Business Process Management (BPM) and CEP for integrating BPM and CEP systems in a closed monitoring and control loop through the exchange of (complex) events. Besides realizing this feedback loop, the proposed event-driven architecture offers a number of further advantages: real-time processing, loose coupling, and scalability. Since models that "work 'on paper' not necessarily work in real world contexts" (March and Smith, 1995, p. 260), we implemented a proof-of-concept using standard software products to demonstrate the implementability and feasibility of the proposed conceptual architecture.

The structure of this paper is as follows: In the subsequent section (Section 2), we present an overview of related work on analytical approaches to BPM and CEP. Based on this literature, we propose a conceptual architecture in Section 3. In Section 4, we focus on the demonstration of the proposed architecture by detailing our proof-of-concept implementation. In Section 6 we discuss experiences made during and after the development and demonstration of this prototype. We close with a discussion of limitations and areas for future research.

2 Technological background

2.1 Business process analytics

Processes are generally defined as sequences of activities performed within or across companies or organizations (Object Management Group, 2009). Activities are considered to be the smallest entity used to build a process. These entities may be seen as complex chunks, which do not need to be analyzed further, or atomic activities, which cannot be divided any further. Moreover, an activity can either refer to work, which needs to be performed manually, or to work, which can be done automatically by a system.

BPM refers to a collection of tools and methods for achieving an understanding of and then for managing and improving an enterprises' process portfolio (zur Muehlen & Indulska, 2010). Davenport and Harris define the term analytics as the "extensive use of data, statistical and quantitative analysis, explanatory and predictive models, and fact-based management to drive decisions and actions" (Davenport and Harris, 2007, p. 7). Analytics may be both an input for human decisions and a driver for fully automated decisions (Davenport & Harris, 2007). Following zur Muehlen and Shapiro (2010), we distinguish between three types of analytic applications in the context of BPM:

Process controlling and process mining. Historical process data can be found in log files of BPMS and other process-aware information systems (zur Muehlen & Shapiro, 2010). This data can be extracted, transformed, and loaded into an external database to allow for a detailed analysis. Typically, process data is analyzed on an aggregated level and from an external perspective (focusing on inputs and outputs of processes, rather than the structure and behavior of a process) (Verbeek, Buijs, van Dongen, & van der Aalst, 2010).

Process mining (van der Aalst, 2011), on the other hand, strives for the "automatic construction of models explaining the behavior observed in the event log" (van der Aalst, Reijers, and Weijters, 2007, p. 714). Its focus is on concurrent processes rather than on static or mainly sequential structures (Verbeek et al., 2010). To its very nature, process mining is an ex-post analysis of process behavior. Process mining poses questions regarding the "process perspective (How?), the organizational perspective (Who?), and the case perspective

(What?)" (van der Aalst et al., 2007, p. 714). Thus, it has the goal of looking "inside the process" (Verbeek et al., 2010, p. 1).

Business activity monitoring. BAM focuses on capturing and processing events with minimum latency, i.e., near real-time (Nesamoney, 2004). BAM foremost strives for transforming process-related events into Key Performance Indicators (KPIs) and for detecting changes or trends indicating opportunities or problems that require managers to take proactive or corrective actions (Nesamoney, 2004). Therefore, a BAM system must be able to detect process-relevant events occurring in an enterprise system, to calculate information for temporal processing, to integrate event and contextual business information on the fly, to execute business rules for setting thresholds according to KPIs and other business-specific triggers, and to provide intuitive interfaces for presenting the results to a user (Costello & Molloy, 2008; Nesamoney, 2004). A typical BAM solution therefore comprises of four components: a processing module, a process definition module, a monitoring module, and a visualization module (Costello & Molloy, 2008; DeFee & Harmon, 2004).

In parts, the BAM concept goes beyond solely tracking KPIs. Some BAM tools contain complex event processing functionality, namely "filtering, transformation, and most importantly aggregation of events" (Etzion and Niblett, 2010, p. 18). In addition, by applying rules engines BAM systems can generate alerts and initiate actions which inform managers of critical situations (zur Muehlen & Shapiro, 2010). "This can be done in batch mode [. . .] or in online mode so that the current value of the KPI can be continually tracked, typically on a dashboard" (Etzion and Niblett, 2010, p. 18).

Simulation and predictive analytics. Simulation is used either before a process is deployed for execution or after a process has been executed for a reasonable amount of time (Rozinat, Wynn, van der Aalst, ter Hofstedde, & Fridge, 2009). In the former case, simulation can assist in assessing the adequacy of the process design for the designated task. In the latter case, variants of a process design can be used for simulation in order to improve the process. In both cases, simulation requires a sophisticated mathematical model of the process at hand, including, e.g., data on the availability of resources and probability distributions for the occurrence of events and the duration of activities.

Predictive analytic techniques, such as time series analysis and forecasting, offer insight into the longer term horizon of process execution and can help identify future bottlenecks in a process or in the process landscape (Bianchi, Boyle, & Hollingsworth, 1999). Time series analysis tries to build a mathematical model that captures the characteristics of time series data, such as a series of measured process KPIs. Time series forecasting, in turn, uses this model to forecast future data points. In comparison to simulation, time series analysis and forecasting requires fewer assumptions about the structure and the behavior of a process, but more historical data.

Currently, approaches are emerging that strive to provide process analytics that helps to evaluate the past, to understand what is happening at the moment, and to predict the future in a more timely fashion, i.e., converging the occurrence of events and gaining analytical insight (zur Muehlen & Shapiro, 2010). Event-based or event-driven Business Intelligence (BI) approaches have been proposed which can run online and are triggered by events. Some vendors refer to such approaches as operational BI systems (Etzion & Niblett, 2010). The ultimate goal of these systems is to facilitate an event-driven decision making. However, the status-quo of event-driven BI falls short when it comes to the dynamic modeling and use of business rules to define actions upon event occurrences and especially in correlating events into metrics over time (Nesamoney, 2004).

2.2 Complex event processing

Complex event processing, in general, comprises a set of techniques for making sense of the behavior of a system by deriving higher-level knowledge from lower-level system events in a timely and online fashion (Eckert, 2008; Etzion & Niblett, 2010; Luckham, 2002).

The central concept in CEP is the event. In common language, an event is something that happens. In CEP, an event is defined as a data object that is a record of an activity that has happened, or is thought of happening in a system (Luckham, 2002). The event signifies the activity. In the context of BPM, events represent "state changes of objects within the context of a business process" (zur Muehlen and Shapiro, 2010, p. 138). Such objects comprise, among others, activities, actors, data elements, applications, or entire processes. Typical events in the context of BPM are, e.g., *process started*, *work item assigned*, or *activity completed* (for more examples cf. Workflow Management Coalition (WfMC)

(2008a)). Attributes of an event can include, e.g., the *ID* of the signified activity, *start* and *end time* of the activity, the system in which the activity has been executed, or the user who carried out the activity (Luckham, 2002).

CEP features the notion of events as nested structures of events. Through the definition of abstraction relationships, it is possible to define complex events by aggregating a set of low-level events. Hence, a complex event signifies a complex activity which consists of all the activities that the aggregation of individual events signified (Luckham, 2002). The complex event *sub-process completed*, for example, is created, when the completions of all activities of the sub-process have been signified by corresponding low-level *activity completed* events.

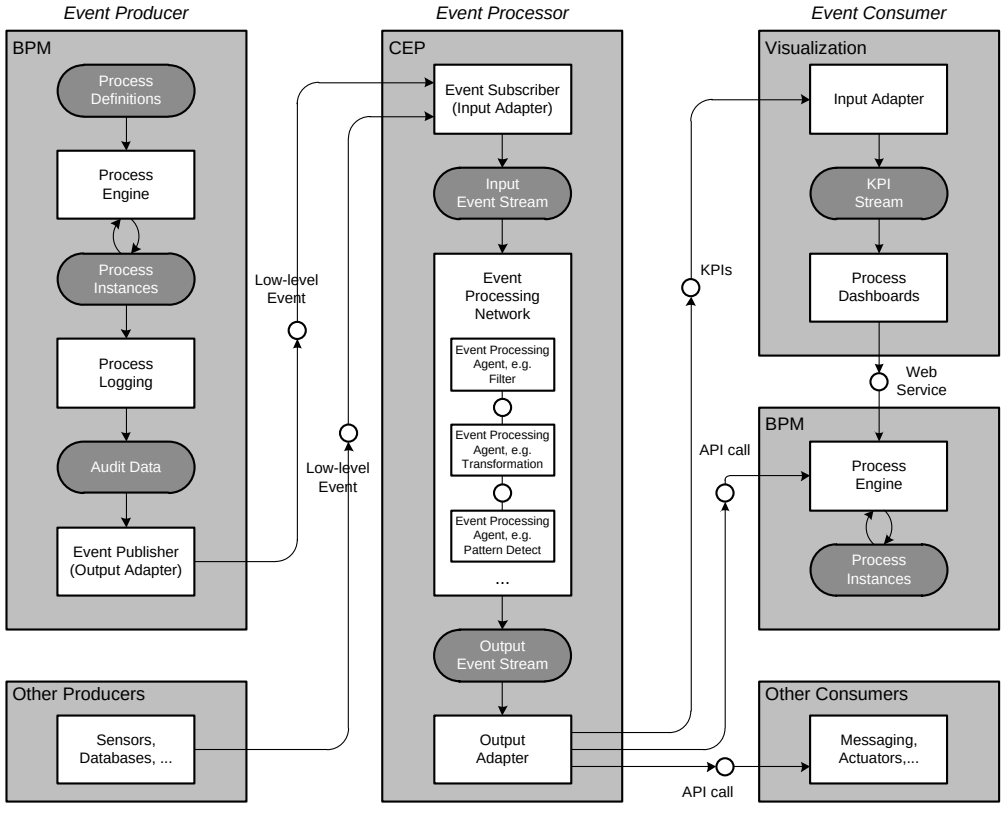
Orthogonal to the concept of aggregation is the concept of causality. Low-level events might happen in dependence of each other (Luckham, 2002). This dependence relationship is called causality. An event is caused by another event if the former happened only because the latter event happened before. In a business process the event order declined might be caused by the event *credit card check failed*. Causality plays an important role in any kind of root cause analysis, both online and ex-post.

CEP emerged as a valuable instrument for a timely analysis of process-related events under the term Event-driven Business Process Management (von Ammon, Ertlmaier, Etzion, Kofman, & Paulus, 2010). Here, a CEP system denotes "a parallel running platform that defines, analyses, and processes events" (von Ammon et al., 2010, p. 371). BPM and CEP systems interact via events. The interaction between both systems features two facets, as a BPMS can act as (a) a producer and as (b) a consumer of events (Etzion & Niblett, 2010). Previously proposed architectures (e.g., Hermosillo, Seinturier, and Duchien (2010), von Ammon et al. (2010)) outline this form of interaction between CEP and BPM, but not yet illustrate which specific software components are required to achieve a loosely-coupled —not hardwired— integration of CEP and BPM. Furthermore, these approaches do not outline how feedback loops, where BPMS act as consumer of the CEP, could be realized.

3 Conceptual architecture

In the following, we briefly describe how CEP can be applied to monitor and control business processes in near real-time. We extend upon the real-time concept of BAM by leveraging the capabilities of CEP to include decision-making and feedback mechanisms to the originating BPMS or other systems. This effectively means that the CEP engine, in parts, actively manages rather than just observes the execution of the business process. Hence, we deem it more appropriate to speak of business activity *management* than of business activity *monitoring*.

Figure 1 depicts the proposed conceptual architecture that is based on the characteristic tri-partition in CEP: It distinguishes between the entities that input events into the actual CEP system (*event producers*), the entities that process the events (*event processors*), and the entities that receive the events after processing (*event consumers*).



cf.: Janiesch, Matzner, and Müller (2011)

Figure 1: Conceptual architecture for event-driven business activity management

Event producer. For analytic purposes, two components of a BPMS are of main interest (for an overview on components of a BPMS cf., e.g., Workflow Management Coalition (WfMC) (1999):

1. The process engine provides an environment for the execution of process instances, based on existing process definitions.
2. The process logging component offers functions to track the progress of single process instances and to log corresponding audit data including data on occurred state transitions of process-relevant objects (e.g., processes, activities, resources, work items).

The resulting audit data constitutes a rich source of low-level process events for subsequent analysis, but is usually not accessible from outside the BPMS. Hence, an additional component – an event publisher – is required to make the data available for external systems. The main functions of this output adapter are sniffing/ detecting, filtering, formatting, and sending of events (Luckham, 2002).

After detecting an event, the corresponding data have to be extracted and transformed into a specific event format. Most BPMS use proprietary formats for log representation. Recently, there have been attempts to standardize event formats, e.g., the XML-based Business Process Analytics Format (BPAF) (Workflow Management Coalition (WfMC), 2008b), the Common Base Event (IBM Corporation, 2004), and the Extensible Event Stream (XES) (Günther, 2009). However, not all system events can be easily mapped into such a standard format. BPAF, e.g., deals only with state changes in processes and activities. Other process-related entities such as applications and work items are not covered. Likewise, not all desired system event attributes might be available or accessible from a system's log. For example, not all BPMS log the relationship between a process instance and an application that is launched by that process instance. Further information might be implicit or is not being persisted in any log and, thus, cannot be exposed in the form of events (e.g., the size of a user's work list). Becker, Matzner, Müller, and Walter (2012) provide a detailed comparison of relevant event formats. The authors show that there is a need for business events and targeted actions. Future research must focus on either merging these approaches with a focus on real-time processing or superseding them with a streamlined format.

After an event has been detected and formatted, a corresponding message has to be sent to the CEP system. Communication can be done in a direct point-to-point fashion or via some kind of message-oriented middleware (for reasons of complexity not depicted in Figure 1). Either way, the use of a publish-subscribe messaging mechanism is a promising way to realize loosely coupled and scalable communication.

Besides BPMS, there are a number of other systems that can act as event sources in the context of a business process. Sensors, for instance, can produce events that signify the real-world execution of a business process activity (e.g., through reading a tag which is associated with a process instance to signify the arrival of a process object such as a delivery truck) or report on real-world events that are happening in the context of a business process (e.g., weather change, traffic conditions, position and movement of objects). Transactional systems, such as ERP or CRM systems, can produce events that are related to a business process but lie outside of the control sphere of a BPMS (e.g., low stock alert, creation of a master data record).

Event processor. The raw events sent out by the various event producers are received and processed by a central event processor, i.e., the actual CEP system.

A typical CEP system comprises three logical components: an input adapter, an Event Processing Network (EPN) consisting of a set of connected Event Processing Agents (EPAs), and an output adapter (Etzion and Niblett (2010, p. 95), Luckham (2002, pp. 40-41)).

The input adapter is responsible for transforming incoming events into the internal event format of the CEP system. Usually, there is an input adapter for each connected event producer and each possible incoming event format. After adaptation, events are queued into an input event stream, ready to be processed by an EPN.

An EPN consists of a number of connected EPAs. An EPA is an active component that monitors an event stream to detect and to react on certain conditions. Its behavior is defined by a set of event pattern rules. These reactive rules trigger on specific instances or patterns of input events. As a result of executing a rule, the agent creates output events and changes its local state. Three basic classes of EPAs can be distinguished (Etzion & Niblett, 2010): A *filter* EPA which performs filtering only, a *transformation* EPA which processes input events, which are defined in its event pattern rules and creates output events that are

functions of input events, and a *pattern detect* EPA which performs a pattern matching function on a stream of input events.

For instance, this capability allows several operations on events concerning the placing of a purchase order. Events could be filtered for specific order IDs, patterns could be detect for duplicate orders in a short time frame, or the events could be transformed by enriching or projecting (i.e., limiting) the information in the event attributes. Also, events can be aggregated to form higher level events (e.g., the sum of all orders of one client) or split up (e.g., one event per order line item).

Having defined a number of EPAs, EPNs can be composed by linking EPAs so that the output of one agent is the input of another agent. This allows for building complex event processing logic out of lightweight and reusable building blocks. So combing EPAs, for instance, allows for filtering for specific products in an order stream first and then projecting only their value and later aggregating this value to have a *current total order value for the day* which is incremented whenever a new order is placed on that day.

Also, EPNs can be composed into even larger networks of event processors for scalability or query distribution purposes. This way, one EPN acts as the producer for another EPN which is the consumer.

After processing, events are queued into an output event stream to be forwarded to event consumers. An event output adapter is responsible for transforming the outgoing events into external formats that can be processed by the event consumers. The data does not necessarily have to represent the original event anymore; it can, for example, be in the form of KPIs, messages, or remote function calls (see below). In terms of KPIs, a stream of order events resulting into a KPI, for example, the total order value for the day could be sent as an event to any subscribed dashboard. Similarly, reaching a threshold for order value could trigger messages to be sent or a remote function call to start a process.

Event consumer. An event consumer is an entity that receives and visualizes or acts upon events from a CEP system. As depicted in Figure 1, dashboards, BPMS, and messaging services are exemplified as event consumers in the conceptual architecture.

"A dashboard is a visual display of the most important information needed to achieve one or more objectives." (Few, 2004, p. 2) Dashboards provide high-level summaries

of information in a concise, clear, and intuitive way and, often, in real-time. Typical information to be displayed in a process dashboard includes KPIs, occurred or anticipated exceptions, or visualizations of the actual flow of process instances.

Apart from being an event producer, BPMS also represent key event consumers in this architecture. Based on the insights derived from processing events produced by a BPMS, an event processor might initiate actions that influence processes in that very system (or other BPMS respectively). An event output adapter of a CEP system might, for instance, make an API call to start new processes or suspend, resume, or cancel running process instances. Likewise, gateways that control the flow of a process instance might be changed by manipulating or updating the conditions of a business rule or the values being evaluated therein.

The last class of event consumers comprises various messaging services that can be employed to send alert messages that inform process owners about exceptions or disturbing KPIs. A CEP engine can basically use all available channels of communication such as mobile phone Short Messaging Services (SMS), instant messaging, e-mail, or mass messaging services such as Twitter to actuate reactions.

4 Proof-of-concept

In order to practically evaluate our conceptual architecture, we assigned a group of seven Information Systems Master's degree students the task to build a proof-of-concept using commercial of-the-shelf software for BPM, CEP, and dashboarding. The students worked on this task within a half-year lasting full-time project called *Slipstream*. We used multiple SAP NetWeaver BPM (SAP AG, 2011a) servers, the Sybase Aleri Streaming Platform (Sybase Inc. 2010), and SAP BusinessObjects Xcelsius Enterprise (now SAP BusinessObjects Dashboards) (SAP AG, 2011c) as well as custom JavaScript as visualization frontends. All of the above software can be downloaded for free for trial and evaluation purposes and represents state-of-the-art BPM and CEP technology. Each of the above software products is used productively in small and medium enterprises as well as in global corporations. However, according to SAP AG, the components have not yet been used together in such configuration.

The resulting technical architecture is depicted in Figure 2. It is derived from the generic architecture illustrated in Figure 1. The elements shaded in dark grey represent custom made components, the elements in light grey standard software components.

The architecture diagram describes the system design from a rather logical perspective, which means that the different elements cannot always be directly mapped to programmatic entities such as Java classes or packages.

The primary source of events is the BPMS, i.e., SAP NetWeaver BPM. However, SAP NetWeaver BPM is not able to send out process-related events by default. To make use of it as an event producer, a modification of the process engine and the development of an additional component were necessary.

The SAP NetWeaver BPM business logger, which continuously logs all activities of running process instances in an audit database, has been patched by implementing a publish-subscribe mechanism for log entries. By doing so, any system component with access to the business log can register itself as a listener for the log and will then be automatically provided with each new log entry object that has been persisted. This listener is part of the Event Publisher, a Java-based extension of the SAP NetWeaver BPM server. It receives all entries from the business log and stores them in an internal queue.

A second sub-component, which is running in a separate thread to decouple its execution from the execution of the BPM server, then further processes the queued log entries. First, different kinds of (configurable) filters may be applied in order to reduce the amount of generated events or to prevent sensible data from being published (an example is to allow only log entries from specific processes to be published as events). Second, XML-based event representations of the log entries (which actually are Java objects) are created. The development team decided to use the Business Process Analytics Format (BPAF) as an event format which has been developed for usage in process analytics and process mining contexts (Workflow Management Coalition (WfMC), 2008a). The design of the Event Publisher allows for an easy integration of other formats. Any other BPMS to join the scenario would need to have a similar event publisher which can communicate in BPAF. Currently, event publishing is not standard functionality but more and more vendors include it in their roadmaps.

The XML-based events are published into an event channel to be used by external systems. This channel is a message-oriented middleware. We used the Apache ActiveMQ

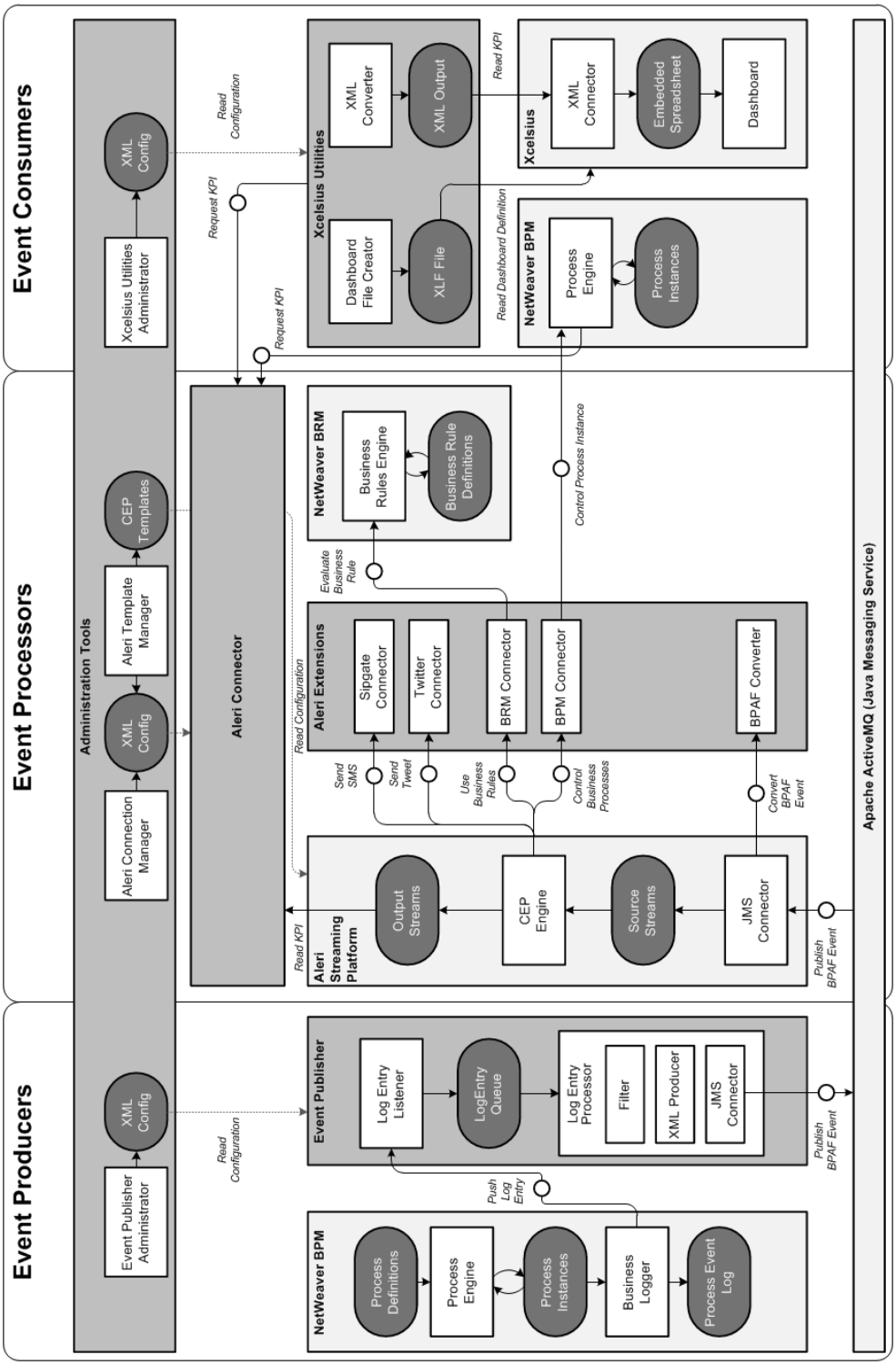


Figure 2: Proof-of-concept of the proposed conceptual architecture

implementation of the Java Messaging Service (JMS), which enables a loosely coupled, but reliable and push-based connection between event producers and the event processor. It also facilitates other systems (e.g., other process engines) to contribute BPAF events to the event stream.

The Sybase Aleri Streaming Platform (Aleri) was used as a CEP engine. To enable Aleri to receive BPAF events, the built-in Aleri JMS connector subscribes to the JMS topic and is continuously fed with events by the JMS broker. As Aleri cannot handle nested XML structures like BPAF natively, the incoming events have to be converted into an internal format (which is basically based on a relational data model). The team implemented a BPAF Converter as part of the Aleri Extensions. The extension makes events available in one (or more) source stream(s), from where they flow through the CEP model, i.e., the EPN) which contains typical CEP operations. The results of the CEP operations are stored in output streams. Any CEP system to be included in the scenario needs a similar converter for XML events. In some cases a simple mapping of attributes may be sufficient, if the CEP engine has a comprehensive XML connector. The team, e.g., swapped the CEP engine (from Sybase Aleri Streaming Platform to SAP BusinessObjects Event Insight) during the project runtime and did not encounter any problems processing the BPAF events.

While executing the CEP models, the CEP engine can make use of further extensions (which are all made available to Aleri via external Java archives):

The Business Rules Management (BRM) Connector enables Aleri to include business rules specified by SAP NetWeaver BRM (SAP AG, 2011b) in event processing. The connection between the two systems is based on Web service calls.

The BPM Connector is used for the control and manipulation of processes managed by the BPMS. With the help of Web services, Aleri is able to automatically start new processes as well as to cancel, suspend, or resume running process instances. These connectors are specific to the implementation as they interact with specific software interfaces. Thus, any new software with different interfaces requires new connectors.

The Twitter Connector uses the Twitter API to enable Aleri to post Twitter messages as a result of CEP operations. The Sipgate Connector provides the possibility to send text messages via the Sipgate SMS gateway. These are two exemplary connectors we implemented to evaluate the ease of adding notification channels.

To make the data which is processed within Aleri available to external systems, we developed the Aleri Connector. It is capable of establishing connections to different Aleri servers and the corresponding CEP models. The actual data can then be retrieved via a subscription mechanism so that registered systems are automatically notified about new events that have occurred in the observed streams. A simple query mechanism that returns the current data in a stream on demand has been implemented as well.

The Aleri Connector is used by two other system components. First, the BPM server itself has the ability to access Aleri data, for example in order to control the process flow in gateways based on CEP results. Second, the measures calculated by Aleri have to be made available to a dashboard such as SAP BusinessObjects Xcelsius Enterprise (Xcelsius) so that they can be visualized.

For this purpose, we implemented the so-called Xcelsius Utilities. One of its sub-components is the XML Converter that uses the Aleri Connector to retrieve calculated values from a configurable set of Aleri output streams. This data is then converted into an Xcelsius-specific XML format and provided as a servlet that can be requested at run-time. The servlet provides the latest values to the real-time dashboarding. The Xcelsius Utilities also comprise an Xcelsius Template File (XLF) Creator which is able to generate template files for Xcelsius. This means that the URL to the XML output servlet and a list of streams whose data should be used for the dashboards are already preconfigured for use. This connector is specific to Xcelsius as there are no standards for push-based updates that both systems could communicate in. Similar issues have been discussed when choosing an event format for BPM events. During project runtime, the team has swapped the visualization for a custom developed graph in JavaScript to Xcelsius. Naturally, the format to communicate with the Web browser had to be adapted, but the effort was low.

The Administration Tools bundle all configuration and administration tasks in an integrated Web portal that has been implemented with SAP Web Dynpro technology. In the background, XML files are used for storing the configuration data. The administration of the Event Publisher module comprises two tasks: On the one hand, the event destinations have to be configured. This includes the specification of connection details for the JMS broker as well as the choice of an event format. Hereby, the system is able to send out events to different JMS destinations using different event formats. On the other hand, event filters can be specified, for instance, event generation can be restricted to events that occurred on

process level and not on task level. All filters can be either defined as a generic filter for all destinations or for a single destination only. Finally, a control mechanism is provided that allows for starting and stopping the Event Publisher.

The second component to be configured is the Aleri Connector. As it is capable of connecting to several Aleri models, which might be running on different servers, the major task is to specify the connection parameters for each model (including hostname, port numbers, and authentication data). The admin interface also provides the possibility to connect or disconnect specific Aleri models.

To enable Xcelsius to consume data from Aleri, the XML Converter as part of the Xcelsius Utilities has to be configured as well. First of all, one has to specify which streams from which Aleri models should be included in the XML output. In addition, different kind of filters can be defined so that only specific columns, only the n most recent rows or only rows where one column contains a specific value are considered. Besides the configuration of the XML Converter, the administration tool of the Xcelsius Utilities contains the possibility to download an XLF.

Finally, the Administration Tools also include the Aleri Template Manager which is able to create, configure, and instantiate CEP templates for Aleri (e.g., templates for standard process metrics and often used KPIs). The templates are XML-based definitions of CEP graphs containing placeholder variables which are then replaced with concrete values for each template instance. The instances can not only be started from the Aleri Template Manager but also be automatically added to the configuration of the Aleri Connection Manager to enable other applications to directly access the data from recently instantiated templates.

While we chose to use specific products for BPM, CEP, and visualization, the communication layer is generic and the team proofed that by exchanging the CEP engine as well as the visualization frontend with reasonable effort. Also, we used standardized BPAF BPM lifecycle events, so any BPMS providing these events could be integrated in the scenario. Due to a lack of interfaces that would expose functionality for taking action on running process instances or starting new process instances, the communication with the BPM engine as an event consumer was less generic.

5 Demonstration

To test the outlined proof-of-concept architecture, we implemented an order fulfillment process (cf. Figure 3; a full technical model is available in the appendix, Section 7) as a demonstrator. The process was provided as a reference process by SAP.

In the showcase the *perfect order fulfillment* —an overall KPI of the process— as well as KPIs representing the performance of individual tasks are measured. A perfect order is an error-free order. Typically error rates are captured at each stage in the ordering process (order entry into system, picking result, delivery duration and method, shipped without damage, correctly invoiced) (Mirror 42 B.V. 2011).

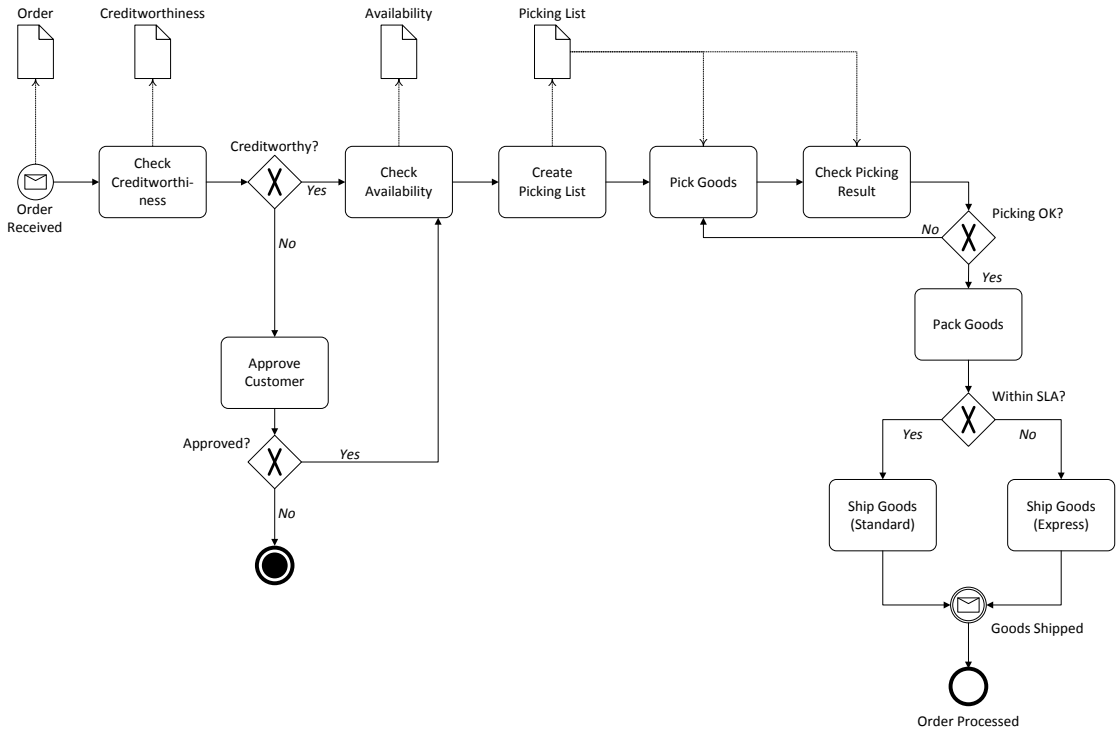


Figure 3: Perfect order fulfillment process

The process model depicts the ordering process[1] at a chip manufacturer who receives orders for certain kinds of computer chips. The manufacturer has to process and ship the

[1] All tasks in the process are Web service calls, the backend ordering system and event sources, however, have been mocked up to simulate different percentages and allow for high volume event demos.

current order within a certain time relatively to the preceding orders to stay within agreed Service Level Agreements (SLAs).

1. The process starts with orders coming into the system. These are processed for certain patterns by filtering and pattern matching: Instances are suspended for manual checks on possible duplicate orders, failed product availability, and creditworthiness checks (only creditworthiness is exemplarily included in Figure 3, cf. the appendix, Section 7, for a full model). Each time such a pattern is detected, the CEP engine sends a suspend process notification to the BPM engine, a work list item is generated for a user to manually approve (and continue the process) or ultimately cancel the process.

2. Once the order has been approved, a picking list is created and the goods are picked from the warehouse. As the picking can be erroneous, we may need to loop back to pick the goods again. Hence, we capture a picking rate for each order. This step is for statistical purposes of the perfect order ratio. It does not have any impact on the process execution except the associated loss of time concerning the SLA.

3. Next, the goods are packed and a shipping method is chosen dynamically while the process is being executed. This is done by choosing an according process path depending on a real-time calculation of the average runtime of the last n instances in order to meet predefined overall SLA. We implemented this by having the CEP engine dynamically set an attribute at the respective XOR gate to choose one of the predefined branches.

4. Once the process has finished, the customer can use a Web form to issue complaints on damaged items which are factored into the overall perfect order KPI (not shown in Figure 3).

The demo was presented on multiple occasions to different audiences. A screencast of the demonstration is available ("Slipstream 2.0: Event-driven Business Activity Management for SAP NetWeaver BPM," 2011). In an internal session the implementation was demonstrated to senior developers and a vice president, the product owner of SAP NetWeaver BPM. In addition, the demo was presented to the public at two SAP TechEd 2010 conferences, the premium technical education venue from SAP, in Germany and the U.S. Overall, the implementation received very favorable feedback. As a result, SAP has planned to integrate

ideas and concepts of our proposed architecture in a future release of its SAP NetWeaver BPM product.

6 Conclusions

Timely insight into a company's value creation processes is of great importance in order to be able to monitor and improve its operations. Business processes are the core building blocks of a company's operations and provide a plethora of information that can be tapped into. However, today business process analytics is often only the second step in BPM. Likewise, near real-time insight into processes is almost non-existent in practice. In the past, custom applications have been developed for this purpose. Recently, the concept of CEP gained attention. However, its dedicated application to BPM is only in its infancy. Nevertheless, it promises process improvements due to the ability to react in a reasonable timeframe to errors, exceptions, or other indicators such as workload, waiting times, etc.

The paper offers contributions to theory and practice. With regard to theory, first, the paper presents an innovative conceptual architecture for leveraging Event-driven Business Activity Management. We achieved this by first reviewing the current state-of-the-art of analytical BPM and CEP. Based on the existing work and the tri-partition in CEP, we developed our conceptual architecture. The architecture is software independent and, thus, can be instantiated with basically any combination of BPM and CEP systems as well as visualizations which are capable of communicating with each other. Through its loosely coupled organization the effort of changing certain components is reasonable and it promises to achieve high scalability.

Second, we reported on our proof-of-concept implementation. While previous approaches reside on a rather abstract level, we consider it a core contribution of our paper that we depict the design and implementation decisions made and actions taken on a very detailed level. We have successfully demonstrated the feasibility of the technical architecture, in addition to Aleri, with a pre-release version of SAP BusinessObjects Event Insight which indicates that —in terms of interoperability— it is possible to exchange the CEP engine at reasonable cost. In our case, it was not necessary to stage multiple CEP engines, but we do not conceive this as a major challenge as we have standardized communication. The implementation is able to showcase the majority of the described use cases. It can be used

to observe and display the state of multiple process engines in a process landscape and to work with information from tertiary systems. As well, it can take action on insight from notification to process control: Processes can be started, suspended, cancelled or altered in their flow. Recently, we have also started to test explorative data analysis techniques to predict future process runtimes in real-time within the CEP engine and to enable preemptive process adaptation.

Third, we have exemplified the implementation in a perfect order fulfillment process which other researchers might use as a demo scenario in order to test their own research artifacts and ideas, such as specific event formats, event producers, and consumers in a realistic environment.

This paper also offers contribution to practice. First, the architecture illustrates a generic approach to achieve immediate insight to action. Organizations might employ this architecture to simply replace custom built legacy applications for process monitoring with a standardized backend in order to make them more maintainable. On the other hand, this architecture can also be used to facilitate a broader vision and include multiple input streams, multiple output options, and multiple processing alternatives for comprehensive BAM.

Second, our proof-of-concept implementation is based on end-to-end software tooling which makes our results more accessible for practitioners. All standard components are publicly available to download and, thus, practitioners can set-up test scenarios and relate the requirements of their specific IT landscape to the proposed architecture based on standard components.

Third, the demo scenario shows a realistic scenario. Organizations might be inspired be seeing the Event-driven Business Activity Management architecture in action. Possibly, it will help practitioners to make sense of potential applications in their organizations. The demonstration exemplified most of our architecture's capabilities in the perfect order use case, which range from mere real-time monitoring to process integration to closed loop automated insight to action.

We did not encounter any substantial problems when implementing multiple BPM engine dashboards. CEP engines accept events from multiple BPM engines without any issues and condensing them on a single dashboard was a mere EPN engineering task. Issues will arise due to load of events, as a BPM engine's logging level can be quite detailed. So a

comprehensive event publisher may be necessary to filter events or the EPN needs to be staged to cope with the event load. But it was our experience that the BPM engine fails first if their event load is too high. We have implemented closed loop functionality which feeds external information about the orders back into the system. With the current setup, it was actually easier to include external data than BPM data as there are often standardized interfaces in the CEP engine for feeds such as RSS. Concerning the feedback of event data back to the BPM engine, we have noticed a severe lack of standards to communicate with the engine. While starting processes was straight forward through Web service calls, other functions such as cancel, suspend or alter were less clear-cut to implement and required changes in the BPM engine.

Table 1 summarizes the process and results of our research project using the design science research methodology of Peffers et al. (2007) as a template.

Next steps will focus on the sophistication of the proof-of-concept and its validation and the transfer of the implementation to other engines. All of this will feed back into the architecture to make it more comprehensive and complete and potentially a reference architecture. In particular, we will create a comprehensive reference scenario which can showcase the abilities mentioned above in one integrated demonstration.

In addition to that, we observed several shortcomings which have to be worked on in the future. Practically speaking, these include but are not limited to the lack of a common business event format and possibly a distinction of life cycle and business events, the lack of a standardized set of process log events, and the lack of standardized operations to trigger BPMS for automated insight to action. On the conceptual side, there is a clear need for an integrated or at least synchronized modeling paradigm for business processes and monitoring applications on the one hand and business processes and their interaction with event processing networks on the other hand. On the vendor side, there is a need for an integrated design paradigm of process KPI specifications inside the BPMS which are linked to an executable CEP model underneath. Otherwise, the dependencies between the BPM and the CEP system will create a significant management overhead which can be prohibitively expensive to maintain.

Table 1: Overview of research process

Phase	Description	Reflection in this project
Identify problem and motivate	The process starts with the identification and justification of the research problem and the motivation of a possible solution	• Current of-the-shelf solutions for BAM or process mining are limited to passive monitoring and may require hardwiring
Define objectives of a solution	Next, objectives of a solution are inferred from the concrete problem definition and knowledge of the general state of the problem and already existing solutions	• Leverage CEP as an enabler of closed-loop process monitoring and control • In addition, CEP provides real-time processing, loose coupling, and scalability
Design and development	At the heart of the process the actual artifact is built. This involves a transformation of the objectives into specific features of an artifact	• Development of a generic conceptual architecture for event-driven Business Activity Management • Development of a proof-of-concept using standard software products
Demonstration	Next, the use of the artifact to solve one or more instances of the defined research problem is demonstrated. Demonstration is done by applying methods such as simulations or case studies	• The proof-of-concept has been demonstrated using a perfect order fulfillment process • The demonstrator has been presented for evaluation to SAP • Ideas and concepts of the proof-of-concept are planned to be incorporated into a future release of SAP NetWeaver BPM
Evaluation	A rigorous assessment using well-defined criteria and empirical data follows up. Methods such as experiments, surveys, or logical proofs can be applied for evaluation purposes	• A formal evaluation has not yet been performed. However, we have positive feedback from practice which fueled SAP decision to include this functionality in their BPM roadmap
Communication	The process closes with the communication of the accomplished results to the scientific and professional community in order to enrich the existing body of knowledge	• Demo has been presented to practitioners and academics on various official occasions

7 Appendix

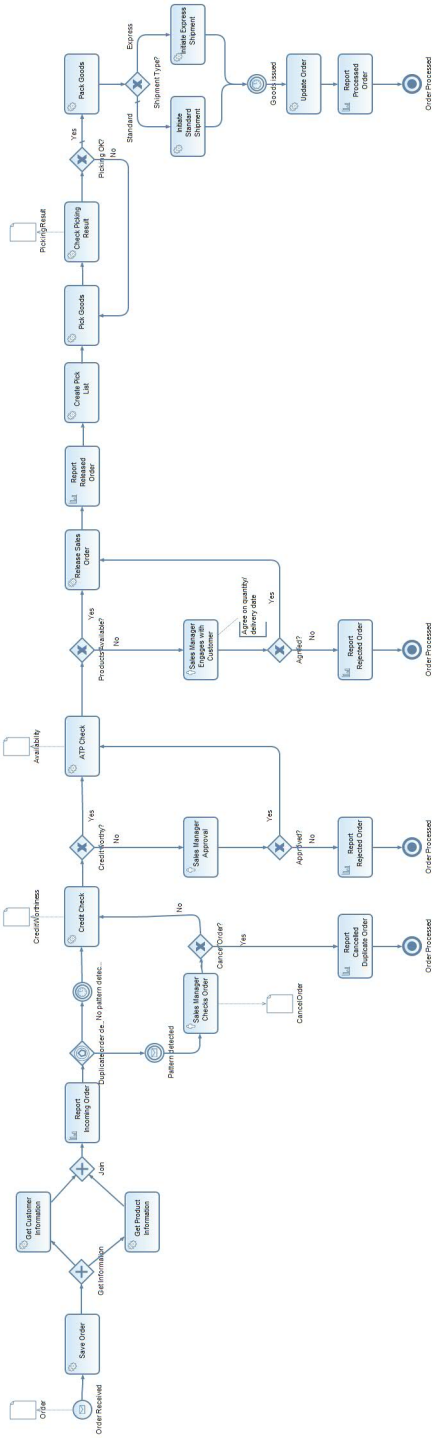


Figure 4: Complete scenario process model

References

Becker, J., Matzner, M., Müller, O., & Walter, M. (2012). A review of event formats as enablers of event-driven BPM. In F. Daniel, K. Barkaoui & S. Dustdar (Eds.), *Lecture Notes in Business Information Processing: Vol. 99. Business Process Management Workshops. BPM 2011 International Workshops* (pp. 433–445). Berlin: Springer.

Bianchi, M., Boyle, M., & Hollingsworth, D. (1999). A comparison of methods for trend estimation. *Applied Economics Letters*, *6*(2), 103–110.

Cheung, M., & Hidders, J. (2011). Round-trip iterative business process modelling between BPA and BPMS tools. *Business Process Management Journal*, *17*(3), 461–494.

Costello, C., & Molloy, O. (2008). Towards a semantic framework for business activity monitoring and management. In K. Hinkelmann (Ed.), *Papers from the AAAI Spring Symposium: AI Meets Business Rules and Process Management*. Stanford, CA: AAAI Press.

Davenport, T., & Harris, J. (2007). *Competing on analytics*. Boston, MA: Harvard Business School Press.

DeFee, J. M., & Harmon, P. (2004). Business activity monitoring and simulation. In L. Fischer (Ed.), *Workflow handbook 2005* (pp. 53–74). Lighthouse Point, FL: Future Strategies.

Eckert, M. (2008). *Complex event processing with XChange EQ: language design, formal semantics, and incremental evaluation for querying events*. (Doctoral dissertation, Ludwig-Maximilians-Universität München, Munich).

Etzion, O., & Niblett, P. (2010). *Event Processing in Action*. Cincinnati, OH: Manning Publications.

Few, S. (2004). Dashboard confusion. Retrieved from http://www.informationweek.com/news/18300136

Günther, C. W. (2009). Extensible Event Stream (XES) Standard Definition. Draft 1.0. Eindhoven: Fluxion Process Laboratories. Retrieved from http://www.xes-standard.org/_media/xes/xes_standard_proposal.pdf

Hammer, M., & Champy, J. (1993). *Reengineering the cooperation: a manifesto for business revolution*. New York, NY: Harper Business.

Hermosillo, G., Seinturier, L., & Duchien, L. (2010). Using complex event processing for dynamic business process adaptation. In *2010 IEEE International Conference on Services Computing* (pp. 466–473). IEEE. Miami, FL.

IBM Corporation. (2004). *Common base event*. Retrieved from http://www.ibm.com/developerworks/library/specification/ws-cbe/

Janiesch, C., Matzner, M., & Müller, O. (2011). A blueprint for event-driven business activity management. In S. Rinderle, F. Toumani & K. Wolf (Eds.), *Lecture Notes in Computer Science: Vol. 6896. Business Process Management, 9th International Concerence* (pp. 17–28). Berlin: Springer.

Janiesch, C., Matzner, M., & Müller, O. (2012). Beyond process monitoring: a proof-of-concept of event-driven business activity management. *Business Process Management Journal*, *18*(4), 625–643. doi:10.1108/14637151211253765

Luckham, D. (2002). *The power of events: an introduction to complex event processing in distributed enterprise systems*. Boston, MA: Addison-Wesley Professional.

March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision Support Systems*, *15*(4), 179–212.

Mirror 42 B.V. (2011). Perfect order measure / fulfillment. KPI library. Retrieved from http://kpilibrary.com/kpis/perfect-order-measure

Nesamoney, D. (2004). BAM: event-driven business intelligence for the real-time enterprise. *DM Review*, *14*(3), 38–40.

Object Management Group. (2009). *Business process modeling notation specification 1.2*. Retrieved from http://www.omg.org/spec/BPMN/1.2/PDF

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, A. S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, *24*(3), 45–77.

Rozinat, A., Wynn, W. T., van der Aalst, W. M. P., ter Hofstedde, A. H. M., & Fridge, C. J. (2009). Workflow simulation for operational decision support. *Data & Knowledge Engineering*, *64*(9), 834–850.

SAP AG. (2011a). Components & tools of SAP NetWeaver: SAP NetWeaver Business Process Management. Retrieved from http://www.sap.com/platform/netweaver/components/sapnetweaverbpm/index.epx

SAP AG. (2011b). Components & tools of SAP NetWeaver: SAP NetWeaver Business Rules Management. Retrieved from http://www.sap.com/platform/netweaver/components/brm/index.epx

SAP AG. (2011c). SAP BusinessObjects Dashboards: see your business clearly. Retrieved from http://www.sap.com/solutions/sapbusinessobjects/large/business-intelligence/dashboards/sapbusinessobjects-dashboards/index.epx

Slipstream 2.0: Event-driven Business Activity Management for SAP NetWeaver BPM. (2011). Retrieved from ERCIS, SAP Research Brisbane: http://www.youtube.com/watch?v=TOdlYJw7WYc

Sybase Inc. (2010). Sybase®Aleri streaming platform tutorial: working with real-time tick data. Retrieved from http://www.sybase.com/files/White_Papers/Sybase_Aleri_Tick_Data_Tutorial.pdf

van der Aalst, W. M. P. (2011). *Process mining: discovery, conformance and enhancement of business processes*. Berlin: Springer.

van der Aalst, W. M. P., Reijers, H. A., & Weijters, A. (2007). Business process mining: an industrial application. *Information Systems*, *32*(5), 713–732.

Verbeek, H. M. W., Buijs, J. C. A. M., van Dongen, B. F., & van der Aalst, W. M. P. (2010). XES Tools. In P. Soffer & E. Proper (Eds.), *Proceedings of the 22nd International Conference on Advanced Information Systems Engineering (CAiSE), CEUR Workshop* (Vol. 592, pp. 1–8).

von Ammon, R., Ertlmaier, T., Etzion, O., Kofman, A., & Paulus, T. (2010). Integrating complex events for collaborating and dynamically changing business processes. In A. Dan, F. Gittler & F. Toumani (Eds.), *Lecture Notes in Computer Science: Vol. 6527. ICSOC/ServiceWave 2009 Workshops* (pp. 370–384). Berlin: Springer.

Weske, M., van der Aalst, W. M. P., & Verbeek, H. M. V. (2004). Advances in business process management. *Data & Knowledge Engineering*, *50*(1), 1–8.

Workflow Management Coalition (WfMC). (1999, April). Workflow Management Coalition Terminology and Glossary. doi:10.1016/S0019-0578(00)00014-8

Workflow Management Coalition (WfMC). (2008a). *Business process analytics format draft version 2.0*. Retrieved from http://www.wfmc.org/index.php?option=com_docman&task=doc_download&gid=437&Itemid=72

Workflow Management Coalition (WfMC). (2008b). Business process analytics format – draft specification 1.0.

zur Muehlen, M., & Indulska, M. (2010). Modeling languages for business processes and business rules: a representational analysis. *Information Systems*, *35*, 379–390.

zur Muehlen, M., & Shapiro, R. (2010). Business process analytics. In J. vom Brocke & M. Rosemann (Eds.), *Handbook on business process management* (Vol. 2, pp. 137–157). Berlin: Springer.