The event processing manifesto

Written by the participants of the 2010 Dagstuhl seminar on event processing

http://www.dagstuhl.de/10201

Dagstuhl Seminar Proceedings 10201
Event Processing
http://drops.dagstuhl.de/opus/volltexte/2011/2985

TABLE OF CONTENTS

PREFACE

The second Dagstuhl seminar on event processing took place in May 2010. This five-day meeting was oriented to work toward a comprehensive document that would explain event processing and how it relates to other technologies and suggest future work in terms of standards, challenges, and shorter-term research projects.

The 45 participants came from academia and industry, some of them out of the event processing field. The teams continued the work after the conference and have summarized their findings in this document. The chapters were written by different teams and then edited for consistency.

The chapter team leaders were: Robert Berry (Chapter 1), Peter Niblett (Chapter 2), Arno Jacobsen (Chapter 3), Paul Vincent (Chapter 4), Bernhard Seeger (Chapter 5), and Patrick Eugster (Chapter 6). The Dagstuhl seminar was organized by Rainer von Ammon, Mani Chandy, and Opher Etzion (who served as editor for this document); the technical editing was done by Sharon Geva.

The following people participated in the seminar:

Rainer von Ammon, Darko Anicic, Stefan Appel, Jean Bacon, Robert Berry, Pedro Bizarro, Andrey Brito, Simon Brodt, Francois Bry, Alejandro Buchmann, Sharma Chakravarthy, Badrish Chandramouli, Mani Chandy, Christoph Emmsersberger, Opher Etzion, Patrick Eugster, Dieter Gawlick, Annika Hinze, Martin Hirzel, Mark Horsburgh, Arno Jackobsen, Boris Koldehofe, Alexander Kozlenkov, Wolfgang May, Daniel Meiron, Ken Moody, Peter Niblett, Adrian Paschke, Udo Pletat, Olga Poppe, Tore Risch, Harold Shcoening, Roy Schulte, Bernhard Seeger, Marco Seirio, Guy Sharon, Plamen Siemonov, Florian Springer, Nenad Stojanovic, John Sutcliffe-Braithwate, Richard Tibbetts, Ronen Vaisnberg, Paul Vincent, Agnes Voisard, Christian Wolff, Carlo Zaniolo, and Holger Ziekow.

All participants contributed to this document.

This role of this document is twofold:

To educate the public about event processing, since it is a relatively new area
To call for action to the community in the areas of standards and further research
One of the highlights of the seminar was the establishment of an event processing grand challenge. We believe that the current applications based on event processing technology just scratched the surface of its potential; the grand challenges offers a focus to make the quantum leap in the impact of event processing on the world.

EXECUTIVE SUMMARY

Event processing (EP) is an area in the field of information technology that is central to many systems on which our society depends. These systems include energy, healthcare, the environment, transportation, finance, services, and manufacturing.

Event processing consists of methods and tools to filter, transform, and detect patterns in events, in order to react to changing conditions, typically under some time constraints.

We present this document to introduce the area of event processing, explain its pertinence to other fields, and to provide information to enable relevant business opportunities. We also aim to establish guidelines for how event processing can fit into current standards and to put forth short- and long-term goals for event processing professionals in industry and academia.

Event processing systems perform the following four main functions:

Obtain data from multiple sources in real or near-real time
Aggregate and analyze this data to detect patterns that indicate the presence of critical situations requiring a response
Determine the best response for such situations
Monitor the execution of that response
Why is event processing of increased importance now, when even the earliest rule engines and business processes had mechanisms used to detect critical situations and respond accordingly?

Today's world is much more dependent on IT systems than it ever was. All of us are much more interconnected and interdependent than ever. Systems must be able to react to events anywhere on the globe. An outbreak of Ebola on one continent, for example, demands a response in countries everywhere. Responses must occur ever quicker, sometimes in milliseconds, as the pace of the stock exchange illustrates.

The costs of inappropriate responses can be staggering, as we see in the cases of certain defense applications. In many telecommunication systems, the volume of data that must be analyzed in near-real time is torrential. In addition, the variety and types of data that must be analyzed in event processing systems is enormous. Such data may be in the form of structured text, natural language, images, audio, or video. The data may be delivered to the system, or it may have to be extracted by the system. In many systems, security is an overarching concern.

Coming decades will see many more applications with event processing capabilities, as society demands smarter ways for managing electric power, water, health, retail and distribution, traffic, and safety—smarter meaning responding better and faster to changing conditions. The interconnected nature of the modern world means that researchers, designers, and students can no longer develop event processing for a single domain, such as the smart grid, without incorporating developments related to event processing technologies in other domains, such as smart healthcare. To step up to these challenges, we are in urgent need of event processing theory, design methods, and tools. This document is an important step toward that goal.

Chapter 1: Why event processing?

In this chapter, we provide an introduction into event processing and describe our motivation for pursuing research in this field. We also provide insight into the value of using an event-driven approach in various systems.

1.1 What is event processing?

Today's information society abounds in myriad information flows, computer-based human collaborations, software agent interactions, electronic businesses, and the explosion of data on the Internet. Understanding what is happening in these environments is becoming increasingly difficult. In other words, we need to find the best ways to make sense of this wealth of data, to improve the quality and availability of information, and to ensure effective responses.

An event is anything that happens [Chandy 2009]. In an information society, flows of data can be seen as streams of observable events. Event-driven systems provide automation to allow the events to be interpreted and correlated, and support the aim of delivering a timely response. Event processing is a set of techniques and tools that help us understand and control event-driven systems [Luckham 2002]. The key idea is to explore temporal, causal, and semantic relationships among events to make sense of them in a timely fashion. This reveals opportunities and threats as soon as they emerge or can serve to diagnose and execute decisions in time constrained fashion.

Most businesses today actively monitor vast quantities of event data to make automated decisions and take time-critical actions. Event processing provides real-time visibility in a wealth of event data and enables responsiveness in decision-making processes. A fundamental characteristic of events is that they cannot be entirely foreseen [Chandy 2009]. Given that fact, we cannot predict when a critical event will happen. What we can do is ensure that the response is provided with minimum latency. Timeliness is one of the essential requirements in many of the event processing applications.

Event processing has emerged as a substantial new field of software engineering and computer science over the last ten years. It is now one of the fastest growing segments in enterprise middleware software, with products provided by major software vendors and many start-up companies around the world. Given this phenomenon, we examine why event processing is emerging now (and not some time before). We then classify categories of problems that can be solved by event processing and consider the typical complexity drivers in each of these categories. We examine the benefits of using this technology and the costs associated with it. Finally, we discuss the buy-versus-build issue to provide some guidelines for successful adoption of this technology.

1.2 Who should read this?

This document is directed toward several key stakeholders in event processing.

In particular, end users should read this to gain insight into the value that the event processing approach provides to solving real business problems. You will learn what the unique characteristics are that make a problem ideally suited to being solved by this approach as well as the costs and benefits of using this approach, as compared to alternate approaches.

We also address the broad applicability of event processing and explore the criteria for and benefits of applying its approach to distinct problem types. Architects and

analysts can use this document to help enable them to recognize cases in which the event processing approach should be applied.

1.3 Origins of event processing

The concept of event processing is older than computing itself. Doing work in response to events, whether it is the receipt of postal mail, the ringing of a fire alarm, or the declaration of war, is a natural part of our society and economy. But recent years have seen a dramatic increase in the number of observable events, in the need for timely responses, and in the automation of those responses. The automation of markets, factories, and communication means that events that were previously inaccessible to computing are now available for processing.

Electronification has also led to people expecting more timely responses to their needs. A six- to eight-week delivery process or a three-day wait for telephone activation no longer meet people's expectations. To achieve timely responses that are acceptable, operations that were previously accomplished by people and paperwork are being implemented with software and electronic communication. Companies today are operating at a faster pace, so the ability to handle more observable events in a short time is increasingly important. The amount of available event data is rapidly expanding as well, because of the decreasing costs and increasing speed of computers and networks and the unifying power of the Internet and its communication standards [Chandy 2009]. The end result of these changes is the emergence of many new systems that are required to process potentially vast quantities of data in a timely fashion and to make complex automated decisions based on the information available to them.

Event processing is experiencing an evolution from bespoke, customized, *ad hoc* designs into an established computing paradigm, just as other technologies have done before. Techniques and computing architectures such as report generation, client-server protocols, and web services all started with customized implementations based on older technology. Over time, best practices and design principles for these systems emerged and were integrated into specialized computing platforms, just as they are now for event processing. Today, a variety of established event processing technologies exist, as do a range of open research problems to be resolved. But as these technologies are improved upon, and problems are explored, many common characteristics have emerged.

Event processing offers a standardized and optimized way to implement event-based applications. While building systems that process events using traditional computing architectures is certainly possible, meeting the requirements of modern systems this way can be difficult. Using event processing, organizations are able to react more quickly and are able to handle more data and more detailed information, thus making better decisions based on more sophisticated logic.

1.4 Problems solved by event processing

To understand why organizations are selecting an event processing approach for implementing advanced IT solutions, we must consider application areas that have already benefited from employing this technology. We must also understand the characteristics of the applications for which event processing technology has been selected.

In application areas like production monitoring and control systems, location-based services, algorithmic stock trading, or logistics control, we can observe a trend away from the traditional client-server interaction model. In this model, the client pulls

information from a server toward a more loosely-coupled, event-based interaction pattern, in which partner applications emit information in an asynchronous push mode.

This shift toward more asynchronous push interactions is supported by an event-driven architecture pattern. This pattern supports event transport and processing as a programming paradigm. This applies to external interactions of a system as well as to system-internal component interfaces. This shift is illustrated in Figure 1.1.

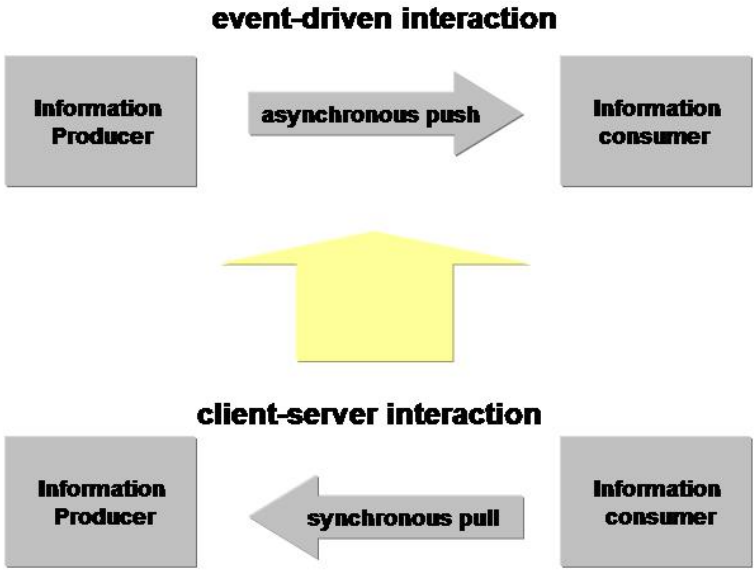


Figure 1.1: Event-driven interaction

A closer examination into the kinds of applications mentioned above reveals the reasons for shifting from request-response mode to event driven mode.

1.4.1 Manufacturing Execution systems

Manufacturing execution systems (MES) are the heart of production processes in various industries. Both discrete and continuous production processes interact with the production machinery directing the production proper. Event processing can be used in MES systems to detect anomalies and determine if significant changes relative to assumptions that require re-planning and predict future events and states have taken place. Event processing in MES systems functions as a subsystem that should interact with the rest of the system. This space of production machinery is an event source emitting condition information about production equipment and processes asynchronously. To enable this interaction, an event processing layer can be found in the majority of MES. This event processing layer is a natural fit for the interface between the manufacturing execution and the plant floor control systems. Using a client-server interaction mode, in which the MES queries production status information from the plant floor, would invert this natural role-split since the MES would become the client for the plant floor control system would logically have to play the server role. An asynchronous push interaction from the plant floor control systems to the manufacturing execution system allows for maintaining the view that the MES is the server, with the plant floor control systems being the clients who feed

information into the MES. The asynchronous push mode from the plant floor systems to the MES appears to be the model of choice. Compared to the client-server pull mode, it also saves one half of the client-server interaction cycle, which allows for better performance of the event-based interaction pattern. Speed of reaction is important in the MES context, as the plant floor systems operate at a much higher speed than the MES. Therefore, a client-server interoperation model between the MES and the plant floor control systems would impose implementing server capabilities in the plant floor systems, slowing them down considerably. So this production execution space nicely shows why an event processing approach is superior to a request/response-based client-server architecture.

1.4.2 Location-based services

Another area for applying event-driven application architectures is location-based services, in which location sensors, such as active radio frequency identification (RFID) tags, mobile phones, and Wi-Fi enabled devices feed information about their spatial location into server-side systems. These location feeds trigger services depending on the spatial location of the client, like notifying transportation status for shipped goods through RFID signals, searching for a restaurant from a mobile phone, or tracking goods in the supply-chain and feeding respective management applications, for example.

Also in these applications, as in the production control space, imagining how the event-driven information push delivery mode from the devices to the servers could be inverted to a client-server based information pull mode is difficult. In a client-server based information pull mode, the servers contact the clients to request location and status information. While in the manufacturing space the production machinery are known event sources, in many location-based services, the sensors delivering location information are not necessarily known a priori to the servers. Thus, server-side location-based service applications typically interact with an a priori anonymous set of clients, which do not necessarily know the servers to which they want to deliver information. This information-delivery-only mode distinguishes event-oriented location-based services from standard Internet applications in which anonymous clients access a server they know in order to retrieve information.

1.4.3 Algorithmic trading

Financial data systems, whether in markets and trading or in surveillance and fraud detection, have dramatically changed over the last 20 years, due to new electronic business processes and increased service expectations of market participants. Many of the new business processes are fully automated through IT systems, for which an asynchronous approach seems to be the most natural—and in many cases the only possible—approach.

Algorithmic trading relies fast decision making based on observations of market activities. Large numbers of market prices for financial products are typically emitted to all market participants simultaneously and with low latency. Successful market participants need to react in (quasi) real-time, using digital trading systems implementing their investment strategies. Therefore, the event-based information dissemination imposes an event processing approach on modern financial IT systems, in which (quasi) real-time market analytics can hardly be implemented in conventional client-server architectures. Recent years have shown that event processing is the right approach for coping with the large amounts of information pushed into the data clouds of financial markets.

1.4.4 Defense intelligence

Over the last 50 years, military intelligence has transitioned from an information-poor environment, in which gathering information was the chief concern, to an information-rich environment. Today, the number of sensors, satellites, and soldiers is pervasive, and the need to present a timely, correct, and integrated view of the information they provide is a critical factor for effectiveness.

Figure 1.2 illustrates some examples of application domains and summarizes the type of functions for which event processing systems are enablers. This is summarized in [Etzion 2010].

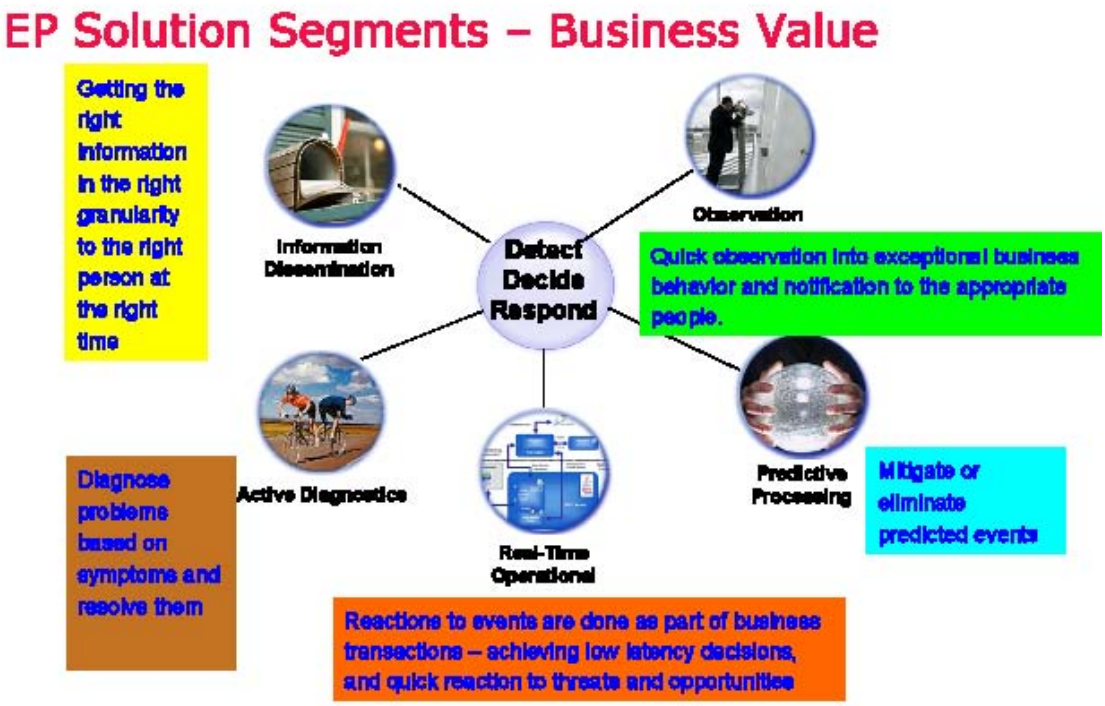


Figure 1.2: Types of functions in event processing

The five different functions from Figure 1.2 are described below, followed by an example of a system in which event processing could perform each function:

Active diagnostics: finding the problem based on symptoms, example: network and system management
Real-time operational decision: reaction to event within time constraints, example: algorithmic trading in financial markets
Predictive processing: predicting future events and proactively mitigating undesired events and exploiting opportunities, example: rerouting due to traffic problems.
Observation systems: observe exceptional behavior (such as KPI threshold breaching) and notify in various ways, example: BAM systems

Information dissemination: subscriptions to particular information that can be filtered, transformed, or derived from streaming events, example: personalized notification from financial data.
A particular application may intend to satisfy more than one of these functions.

1.5 Criteria for adopting an event processing approach

An event processing approach is ideally suited for applications delivering situational awareness and response, with a combination of the following requirements:

The system(s) under control/observation is event-driven, with a high volume of events representing accumulating, incremental state change
Timely action is required—in some cases, that action must have the potential for immediate effect
The application has a high degree of complexity as measured by any of the following factors:

1. Degree to which the application is expected to change over time, e.g., with new event sources, new interactions and new responses expected
2. Numbers and types of event sources
3. Numbers of consumers of information communicated in the events
4. State and context management
5. Opportunity to create new value, e.g., by introducing reflection and introspection

The event data can be made available in electronic form
There is a continuous evolution of requirements, e.g., new event sources/types or new response opportunities
Some applications that do not meet these criteria may also benefit from this approach, but the benefits might be less significant.

Event volumes and timeliness are two factors that most typically influence the selection of an event processing approach. However, other complexity factors can motivate this decision as well. In Table 1.1, we summarize this in a simplified decision table that shows areas in which event processing can be particularly applicable.

In this table, event rates refer to the rate of input events to the systems. Application complexity refers to the five criteria listed above, and timeliness refers to the level of importance of the reactions meeting time constraints.

Event driven nature	Event Rates	Application complexity	Timeliness	Recommended for event processing	Examples
High	High	High	High	Yes	Advanced algorithmic trading
Medium/ High	High	High	Low	Depends on the type of complexity and the secondary benefits of improved timeliness	Supply chain management (SCM), fleet management
High	High	Low	High	Yes	Order routing
Low/Medium	High	Low	Low	Limited event processing capabilities such as filtering may be sufficient	Possibly suited to business activity monitoring
High	Low	High	High	Yes	Earthquake detection using distributed low cost sensors (low average rate, high bushiness)
Low	Low	High	Low	No; Compute-heavy transaction	
Low	Low	Low	High	No; Use messaging system	
Low	Low	Low	Low	No	

Table 1.1: Event Rates, timeliness, and other complexity drivers

Event-driven applications sometimes need to process large volumes of data and enable appropriate reactions using sophisticated logic. The following measures of complexity also motivate an event processing approach:

Degree of importance of timeliness: The effectiveness of a response depends on its timeliness. For example, a tsunami warning issued after a tsunami strike has little value. Event processing is a valuable technology in applications in which the value of a response to an event decays with increased response time. The more rapid the decay, the higher value of the technology. Certain applications (e.g., algorithmic-trading applications) would not exist today if a high-value response was not related to timeliness. On the other hand, event processing is also valuable in applications in which timeliness is not the most critical complexity driver. Therefore, examining other complexity drivers is also worthwhile, as discussed below.

Intensity of event data volumes: Event processing provides intelligent analysis of streaming data (events) in a timely fashion. As the amount of available data in digital form is rapidly increasing, the challenge becomes yielding the intelligent analysis on larger volumes of data. Event processing offers concepts and tools that provide on-the-fly analysis on large volumes of data.

Frequency of application requirement changes: In a typical lifecycle of a business application, its requirements frequently change. The changes happen due to the need to meet new customers' requirements or new government regulations. For whatever reason, business requirements change, and the complexity of the application may also increase with these changes. Event processing systems offer a high-level abstraction layer, in which changing an existing event pattern or adding a new one may significantly influence businesses strategy. Hence, frequently changing event patterns provides a means to cope with frequent requirement changes.

Number of event types: Handling interactions among different parts of an application, or interactions among distributed applications may be a tedious task. Communications of one-to-one or one-to-many may be also hard to manage. In event-driven systems, different interactions and communications are modeled with different event types. By publishing definitions of event types to all interacting components, we can effectively increase transparency of the communication and ease the application integration. Moreover, we improve manageability of complex applications. We can also respond to rapid changes of interactions by changing event types.

Number of consumers of events: Publish/subscribe is a messaging paradigm in event processing, in which senders (publishers) of messages are not programmed to send their messages to specific receivers (subscribers). Publishers are loosely coupled to subscribers and need not even know of their existence. This mechanism may therefore enable greater scalability in applications in which a large number of event consumers (subscribers) exist, as well as in applications in which a more dynamic network topology among consumers is required.

State and context management or rule interdependencies (including complexity of queries): Real-time applications with many interactions are difficult to understand and program. Event processing is a set of techniques meant to help us understand and control such systems. It provides abstractions to cope with these complexities. These abstractions are used for capturing business situations that we want to monitor. That is, they enable a business user to express high-level business goals, hiding away the complexity of the event-driven interactions.

Opportunity for the introduction of new value, e.g., reflection, introspection, or retraction: Having an event as a first class citizen in a real-time programming model opens new horizons and challenges. For example, we can make new-generation applications dealing with the following: inexact event processing (i.e., uncertain data

stream processing), out-of-order event processing (i.e., processing of delayed events), introspection (i.e., inspecting why a certain business event happened and why another, expected one, has not), event retraction (i.e., retracting erroneously recorded events and computing revisions on detected complex events), and so forth. In general, the presence of these factors renders the event processing approach beneficial. Event processing has evolved to address these kinds of challenges. Tools, analysis, and modeling methodologies have been developed, and together with event processing run-time environments (middleware), a compelling case for event processing is emerging.

1.6 Benefits

Enterprises use event processing to improve the performance of their business in many ways, including increasing revenue, lowering costs, reducing risk, and complying with regulation. The essential role of event processing is to provide information that leads to faster and better decisions, as compared to decisions based on the use of traditional applications, management reports, or business intelligence (BI) initiatives. Event processing enables sense-and-respond behavior, in which incoming information is used to sense the current situation and a person, device, or automated component responds in a timely fashion.

1.6.1 Event processing systems provide information faster

Such systems act on a new event as soon as it arrives. The receipt of event data triggers computation immediately, in contrast to traditional applications and business intelligence systems, which are either time-driven or request-driven. Time-driven and request-driven systems store data when they arrive, and processing is triggered later by a clock (in a time-driven system) or by a request from a person or computer program (in a request-driven system). This postpones the computation and therefore defers the response.

Event processing software minimizes processing time by keeping most data in memory and using data structures that are designed to facilitate fast access. Many calculations are incremental. Event processing systems also implement other strategies to minimize excess code path, context switches, and other overhead.

1.6.2 Event processing systems improve the quality of available information

Event processing systems combine information from many data points to generate new insights. They also use algorithms and rules to process event data that they have received from one or more sources during some time period. The systems also generate summary-level facts (complex events) and put them in context to identify threats and opportunity situations. Event processing systems apply a variety of techniques that are particularly appropriate for handling event data. These techniques include provisions for dealing with time windows, such as events that occurred within a few minutes of one another, or events that occurred within the most recent few seconds, minutes, or hours. Event processing systems have ways to identify events that cause other events to happen (causality).

As a rough guideline, alerting systems, reports, and dashboards that must recalculate in ten minutes or less are likely to require event processing approaches. Applications that run less frequently can generally be implemented by traditional, periodic IT architectures. However, even some of those applications would run faster, consume less computer resources, and be easier to develop if they were

implemented using continuous event processing, particularly if they deal with temporal or causal relationships among the event data.

1.7 Benefits of event processing systems

The speed and information-generating capabilities of event processing systems lead to four kinds of benefits:

Operational changes can made sooner, making them more effective
Radically new, time-sensitive business practices are made possible
Better information leads to better decisions
Staff cost is reduced by offloading parts of work to computers
These are explained in the subsequent four sections.

1.7.1 Operational changes can be made sooner, making them more effective

In many aspects of company operations, managers can achieve an economic benefit by fine-tuning the way work is done to more closely match current conditions. Companies reduce their costs or increase their revenue by implementing management decisions earlier, rather than making the same or similar changes later (see Figure 1.3).

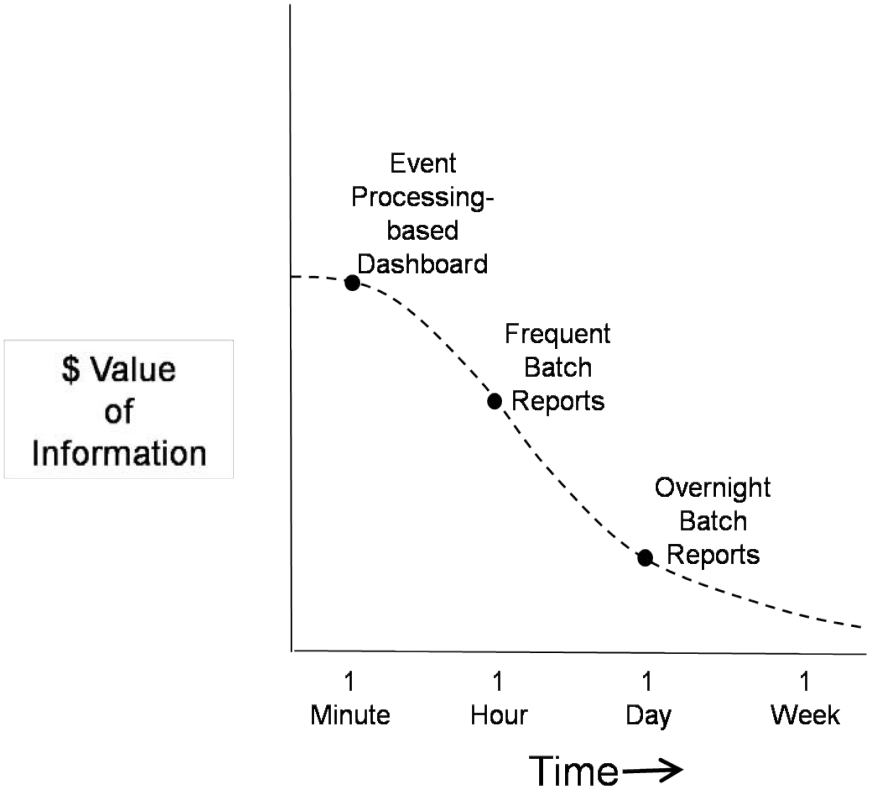


Figure 1.3 Value of having operational management information sooner

For example, customer contact centers were traditionally managed by means of daily reports. These reports provided statistics on call volume, customer on-hold wait time, call duration, transferred calls, dropped calls, and whether the customers' issues were resolved in the first phone call. Contact center managers and supervisors used the reports to decide whether to bring in more agents, adjust staff

assignments, modify the call scripts, or change other aspects of their day-to-day operations. Over time, contact centers began moving to hourly reports to enable more frequent, intra-day adjustments, rather than making the changes on a daily basis. If call volume and wait times grew, agents on outgoing calls could be switched to incoming calls, remote agents could be added to the available work force, or other aspects of the operation could be modified. Idle agent time was reduced and customer service levels were enhanced, but further improvement was still necessary.

Today, many contact centers have moved to continuous monitoring, based on event processing. Sophisticated customer relationship management (CRM) analytic suites incorporate real-time dashboards with graphical displays that update every minute, providing near-real-time visibility into the state of the contact center. Supervisors can spot problems as they emerge, analyze root causes, and quickly make changes. This minimizes wasted time and staff cost and maximizes customer service and long-term company revenue. If data indicate that an individual agent has had several problematic calls in a row, a supervisor can be alerted to give the agent an early break to recover.

Other examples in which event-based monitoring systems improve the timeliness of operational management decisions include airline operations and supply chain management systems.

1.7.2 Radically new, time-sensitive business practices made possible

Event processing enables some radically new business processes that were previously impractical because the necessary information could not be obtained in a timely manner. Event processing systems can collect and analyze a large body of data relevant to a current situation fast enough to affect individual transactions still in process.

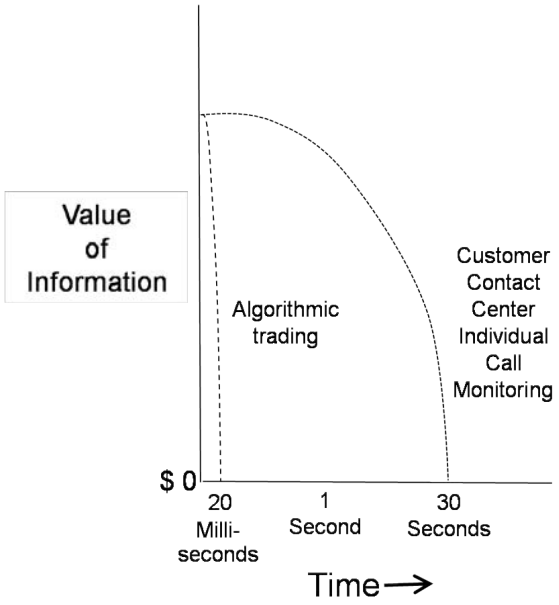


Figure 1.4 Value of information to affect individual transactions

For example, agents and supervisors in a customer contact center can use information derived from event data to adjust their treatment of a customer while he is still on the phone. An agent dealing with a customer who has been transferred several times or has been on hold for more than five minutes can be prompted (and allowed) to give additional compensation for the negative experience. The contact center software may also generate screen pops, messages that prompt agents to ask customer-specific questions to drive up-selling or cross-selling.

Alternatively, a supervisor could become involved in a call if an agent is struggling. Systems must generate alerts within a few seconds if the goal is to enable intervention in a call that is still in progress (see the line plot on the right in Figure 1.4). After 30 seconds, the window of opportunity to affect that transaction has closed.

Similar kinds of improvements are found in fraud-detection systems, customer experience management systems in gambling casinos, and mobile telephone provisioning systems.

High frequency and other algorithmic trading systems in capital markets are even more time-sensitive. Some trading strategies require that buy and sell decisions be made within a few milliseconds, because the opportunity would be gone when prices change slightly, or when a competitor has grabbed the deal. In such cases, merely performing event processing well is not enough—the company must do it faster than its competitors. A particular calculation can be worth hundreds of thousands of dollars if action is taken within a few milliseconds but be worthless 10 or 20 milliseconds later (see the line plot on the left in Figure 1.4). These systems must be fully automated; involving a person in an individual trade could not be done in time. In fact, a person takes 100 milliseconds just to type one character into a keyboard. Similarly, to be effective, some military weapons and defense systems must also operate using such extreme response times.

1.7.3 Timely information dissemination leads to better decisions

The information provided by event processing systems is often impossible or impractical to obtain through any other means. Event processing systems can quickly extract and distill the information value from dozens, thousands, or even millions of data points. From these data, EP systems can produce alerts, key performance indicators, or other metrics to aid in decision making. By contrast, a person, without the assistance of EP systems, can directly assimilate only a few data points at a time, and thus cannot consider nearly as many factors when making a decision.

For example, companies that manage large fleets of trucks get raw information about the location of all of their trucks through GPS systems. Basic fleet management systems can display truck locations on maps so dispatchers can know the location of each truck. However, such massive volumes of data can make spotting situations that require attention difficult. Event processing systems can easily compare each truck's location against its planned itinerary and raise alerts for the few trucks that are far from their expected locations, or for trucks that are within minutes of being ready to load, unload, or require some other kind of attention.

Event processing systems are typically used to implement management-by-exception strategies. They reduce the volume of unwanted data, known as information glut, presented to people. In some cases, a system may run for hours or days, turning millions of base data points into thousands of complex events before detecting a single complex event that must be brought to the attention of a person. People are

disturbed less often, so they can reserve their attention for the few situations in which their involvement is important.

The need to deal with high volumes of current event data is emerging in many applications and industries, including the following examples.

Track-and-trace systems can report the history of pharmaceuticals from factory to patient to combat theft and drug counterfeiting and to conform to regulations. Such systems may save detailed event history data for millions of packages, for months or years. They can retrieve and correlate the data when a question about a particular item or a batch of goods is raised.

Context-aware, location-based services in mobile telephone networks use event processing to track the locations of hundreds of thousands of subscribers simultaneously. The system can notify a person if someone in her group is nearby, by finding matches among millions of data points.

Network security systems use event processing to detect patterns that indicate denial-of-service attacks or various forms of fraud, such as poaching and spoofing user IDs or theft of service.

Package delivery services scan millions of packages several times per day using bar code or RFID readers to monitor and report the history, current location, and projected delivery time of every box or envelope.

In all of these examples, the information value in each individual data point may be small, but the value of accurately distilling a large amount of data into a few significant, complex events is enormous. Computers do not succumb to boredom, so they make fewer mistakes than a person (even given that he had the time and ability to do the computations). Computer calculations are consistent and repeatable, because they always implement the same algorithms in the same way, as defined by knowledge workers, analysts, and software developers.

1.7.4 Staff cost is reduced by offloading work to computers

Some applications can be done fast enough and well enough by humans, but can be done at a lower cost by event processing systems. EP systems offload the drudgery of repetitive calculations and pattern detection comparisons from people to computers in many scenarios, including in the following two examples.

The spread of algorithmic trading systems has reduced the number of human traders operating on certain categories of investments in capital markets. Traders still formulate the strategies and rules that are executed by the trading systems at run time, and traders also still carry out more complex, dynamically-determined trades, and monitor the performance of the automated systems.

The number of people monitoring the operations of airlines, shipping fleets, and trucking companies has also been cut in some areas, because EP systems enable each person to track more vehicles.

The amount of decision making that should be offloaded to EP systems varies depending on the business problem. Managers and knowledge workers can delegate as few or as many tasks as they see appropriate to the software.

1.8 Build-versus-buy

Build-versus-buy remains a major issue in the event processing market. Any event processing application can be implemented with standard programming languages and tools, and before the emergence of commercial event processing platform

products, all applications that needed EP capability used application-specific (build) code to do so. Build EP logic was still used in about 95% of EP applications in 2010, because most event processing applications have relatively infrequent changes and modest requirements regarding throughput, latency, and rule complexity. However, this is changing, as organizations begin to recognize the non-linear rise in costs as change must be introduced, and begin to appreciate the competitive advantage that timeliness (and other differentiators) can deliver.

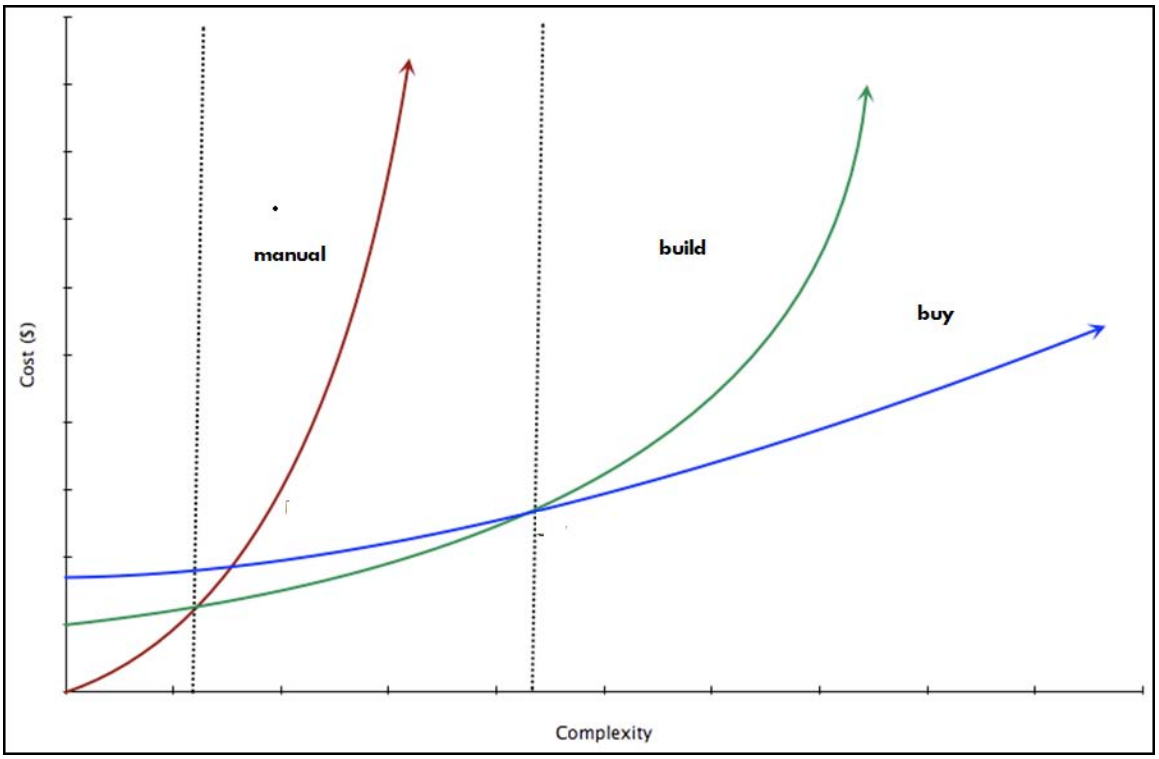


Figure 1.5. Cost vs. complexity in use technologies

This tradeoff can be considered in light of three stages of maturity—manual, build (or adapt), and buy (e.g., using existing EP technology), as illustrated in Figure 1.5.

The curves represented in this figure are actually quite generic and can apply to any new technology and accompanying services capability. The purple curve represents a manual approach, meaning without the use of EP systems, and the green curve represents the build option, while the blue curve represents the buy option. The curves reflect the non-linear increase in costs associated with rising complexity. They acknowledge that adopting a manual approach for simple applications (hiring more people to monitor telephones or computers, for example) might be easier than adapting an EP system, but that at some point, the introduction of new challenges will render a manual approach unaffordable.

Such challenges could be in the form of higher event rates or more event types. A build approach might be considered as the first upgrade from a manual system, in which custom software is developed, but again, the complexity of managing that stack and the probable lack of flexibility to address new opportunities will eventually have the same result. An enterprise should consider moving to the next phase when the complexity rises and makes the upgrade cost-effective.

We believe that more demanding, high-volume/low-latency applications are increasingly likely to use the buy approach, namely by employing commercial-off-the-shelf (COTS) event processing technology

Table 1.2 illustrates some considerations in the build-versus-buy decision.

	Build	Buy
Pro	• Purpose-built • Simplicity • Lower learning curve	• Lower time to benefit • Greater flexibility • Time to change is lower • Existence of Support/Tools/Services
Con	• Cost of development if the application is complex • Lack of flexibility • Higher risk • No services available	• Cost of acquisition if the application is simple • Cost of integration to existing systems • Vendor risk

Table 1.2: Build-versus-buy table

1.9 Summary – the value of event processing

This section surveys some considerations in answering the following questions:

When does it make sense to use an event processing approach?
In which cases does using generic event processing software suffice?
The main factors that support using the event processing approach are:

Use of an application that includes some event-driven requirements (a response is required following certain events)
The applications event-driven requirements are relatively complex
Timing constraints apply to the required responses
Complexity is often the crucial factor, since satisfying requirements using conventional programming abstractions may be difficult. But in some cases, the timeliness of reactions is an important consideration as well.

Chapter 2: What are the characteristics of event processing?

This chapter examines more deeply what we actually mean by the term event processing. In the following sections, we expand on our general definition of the term and address the kinds of problems that event processing can be used to solve. We then describe a set of characteristics common to these event processing applications.

2.1 What do we mean by event processing?

We start with some terminology.

Event processing: This can be broadly defined to be any computing that performs operations on events. Common event processing operations include reading, creating, transforming, and deleting events. They can also include the distribution and dissemination of events among participants in a distributed computing system. The term can refer to specific software or hardware that does the processing, or to the subject area as a whole. Processing of events is usually done with the purpose of meeting some kind of application goal, leading to our next definition.
Event processing application: This is an application of event processing to solve a particular problem, or more generally, any computer application that embodies principles of event processing.
Event processing platforms: These refer to systems and tools that help develop, run, manage and maintain event processing applications. These are frequently used to take some of the burden of programming event processing off of the author of the application.
Event processing involves two main ideas:

Processing events to gather meaningful or valuable information and then deriving actions from them:
The events in question usually represent some aspect of something that occurred in the real world. The significance often arises from the combination of many events, rather than any single event, and time is often important. The information conveyed in a set of events often depends on the order in which they occurred, and/or the time at which they occurred

The value of information may well depend on the timeliness with which it can be supplied to a consumer, varying from consumer to consumer and depending on external circumstances, such as the state of the overall system.

Disseminating or distributing the events:
This idea refers to capturing events and then transporting them to the consumers who need them, ensuring that the appropriate intermediate processing is performed on them. Disseminating and distributing is also about getting the right information to the right consumers at the right time.

The distribution process serves to decouple the event producers and the event consumers, so that event producers do not need to be aware of the physical addresses/identities, or even the existence, of the consumers. In many cases, the consumers are not concerned with the identities or addresses of the producers. This decoupling (often achieved using a publish/subscribe paradigm) allows event processing applications to evolve over time, in a flexible way. Further event

producers, consumers, or event processing functions can be added without disrupting existing users or functionality.

As well as transporting event messages, an event distribution system might need to provide adapters. Adapters allow other applications to act as event producers or consumers. Distribution systems also frequently provide mechanisms to advertise the availability of events or to allow consumers to register their interest in particular event types.

A large scale event processing application might involve thousands of event producers or consumers, dispersed over a wide geographic area, as in a track and trace application monitoring a supply chain, for example. Such an application might have to handle tens of thousands of events per second. So some applications need an event distribution system that can scale to handle high event volumes and large geographic areas, and possible to also span multiple types of communication technology—private networks, public internet, wireless mobile networks, or wireless sensor networks, for example.

2.2 Characteristics of event processing applications

The rationale for event processing is that it facilitates the design and implementation of a certain class of computing applications, which we call event processing applications. So what exactly is an event processing application?

First and foremost, these applications are centered on the ideas of events and actions. Such event can be ingested from the outside world, generated by the application itself, or derived from the processing of these events. Applications are frequently event-driven, in that events are pushed through the system and are processed as soon as, or soon after, they occur. In some cases, events may be stored in a database or log for subsequent retrieval and further processing, but the initial processing is still immediate.

The way in which events are processed in an event processing application is influenced by the context in which the events occur. In many applications, the temporal context—either the absolute occurrence time, or the ordering of events with respect to one another, or both—is very significant.

As we noted in the previous section, event processing applications may involve large numbers of event producers and consumers, which are often distributed over many systems. These systems might have differing communications capabilities and might be dispersed into widely different geographic locations. An application may also need to distribute its processing of events over multiple machines to achieve throughput and latency objectives. As such, event processing applications can be heavy users of concurrent processing techniques.

Some event processing applications are highly dynamic. Event volumes may vary over time, and the nature of the processing may vary depending on the interests of the consumers or the nature of the incoming events. In addition, event producers and consumers may come and go during the execution of the application.

2.3 Event processing application requirements

We now enumerate the main functional capabilities required by event processing applications. Event processing applications come in many shapes and sizes, and as such, not all applications will necessarily use every capability listed in this section. Also, a capability can be implemented either natively by the application, or provided for that application by an event processing platform.

2.3.1 Event Input/Output

Most applications need some way to ingest events from external event producers. These could be hardware sensors, software probes, external data streams (such as stock prices) or news feeds, event logs, or files of pre-recorded events.

Similarly, applications need a way to output events to external consumers. This can be performed by actuators, business applications and processes, and event logs, for example.

2.3.2 Data Reduction

The process of extracting information from events often involves techniques that reduce the amount of data contained in the ingested events. These techniques include:

Filtering: This refers to eliminating entire events as un-interesting, either by nature of the event or according to some other metadata associated with the event (such as its producer) or to the values of attributes contained in the event. Some applications use sampling filters, which select a subset of incoming events, to achieve a particular data rate without regard to other criteria.

Projection: This means reducing the size of an event by discarding some of its attributes (for privacy reasons, for example).

Aggregation: This refers to combining a set of events by extracting one or more attributes from each and performing some kind of computation on these attributes (averaging or calculating the minimum or maximum, for example).

2.3.3 Reasoning:

The techniques used to extracting derived events out of the raw events include:

Transformation: This is the process of taking an event and producing a new one that presents more meaningful information. This information can be derived by analyzing the data contained in the original event (such as car license plate recognition), or it can be derived with the help of some external data source (a database of customers, for example).

Aggregation: In addition to their role in data reduction, aggregation operations can be used to derive additional information (counting event arrivals, for example) or to eliminate errors and noise from incoming events.

Pattern detection: This is a type of processing that examines a collection of events and looks for matches or other patterns among them. The presence of such a pattern often signifies something of more importance than the raw events themselves. A special case of pattern detection is absence detection, in which the fact that a given event *did not* occur in a particular time period is of itself significant.

2.3.4 Context awareness

The way in which events are processed, including the data reduction and reasoning operations we have just mentioned, often requires taking into account the context in which the events occurred. This context can involve one or more of the following:

Segmentation using A key attribute (or combination of attributes) of the event; This is frequently used to identify an external entity to which the event relates, for example, the ID of a patient wearing a health monitor, or the stock ticker symbol in a trading application. We refer to this as a segmentation context.

The state of an entity external to the event processing application: This could be an entity referred to directly by the event. We would expect, for example, a person's heart rate to be different depending on whether she is resting or exercising. This could also refer to something more general, such as the weather conditions where the event occurred or the state of the business in a business-oriented application.

The location at which the event occurred, or its proximity to other related events

The time at which the event occurred, or its temporal relation to other related events

2.3.5 Logging and analysis

Applications frequently keep a record of some or all of the events they receive and of the derived events they generate. This can be used for audit purposes, for retrospective event processing (revisiting precursor events when examining a problem, for example), or to allow offline analysis.

Logging: A history store keeps a log of events that can be later queried.

Provenance tracking: This extends the history store to include information about how each logged event came to be produced. In particular, if the event was one that was derived by the event processing system itself, this information could include the events and derivation process that gave rise to them.

Visualization of event data: Most event processing applications include some kind of visual display of raw and/or derived events. This can be the primary purpose of the application, as seen in many monitoring or customer-facing track and trace applications, or it can be a tool used in development, testing, problem determination, or operational management. Business activity monitoring and network and system monitoring tools provide sophisticated dashboards for display of visual information. Map-based mashups are a popular way to display location-specific events.

2.3.6 Prediction

Event processing applications are sometimes used to make predictions about the future.

Predictive pattern detection: This can be used to look for specific patterns that indicate when an event is likely to occur. In particular, applications can be used to detect anomalous patterns that suggest a problem is likely to occur. For example, a traffic management system that could spot that gridlock is about to occur is more useful than one that can only detect that it has already occurred. The patterns used (and any parameter thresholds contained in them) have to be carefully chosen to avoid too many false positives or false negatives.

On-line scoring: Events, or collections of events, are pre-processed using the techniques we have already addressed and are then passed to a scoring engine, which runs them against an appropriate data-mining model. This process can be used to assign a probability rating to the occurrence of a future event.

Simulation: Another way to predict the future is to use a computer model that simulates the domain being studied and to feed real-life events (after some pre-processing) into that model.

2.3.7 Learning and Adaptation

In many applications the operations performed on events are specified by human beings, either by the programmers responsible for designing the application, or by users of the application who supply rules specifying some or all of the processing to be performed. However, machine learning and similar techniques are being increasingly used for event processing, particularly in applications used for making predictions.

Pattern discovery refers to the automated generation of the event patterns to be looked for by subsequent pattern detection. One approach is to run offline analytics against a log of stored events

Building a scoring model: Applications that use online scoring need a model for that scoring. These models can be built by feeding them events from the event processing system and can be adapted over time by feeding back predicted and actual results. This will reduce the incidence of false positives and false negatives.

Application evolution: Event processing applications often must evolve over time. The volume and the nature of raw input events available to the application can change. In addition, the required processing, such as the patterns being detected, can vary—either in response to new user requirements, or in response to situations being detected by the EP application itself.

2.3.8 Distribution

Last, but not least, event processing systems must ensure that events are transported from the places where they are detected to the place or places where the event processing occurs. EP systems must also ensure that any outcomes from that processing are delivered to the right places.

Routing: This means ensuring that incoming events are routed to the right processing logic, and that output events are routed to the consumers of those events. In simple cases, this routing is static, meaning that the routes are defined when the application is developed and only change if the application is explicitly modified. In contrast, dynamic routes can be created, deleted, or modified while the application is running, either as a result of user interaction or as a result of situations detected by the application itself.

Partitioning: Event processing can itself be distributed across multiple servers. One common approach is to partition the work across a horizontal cluster of servers and to route events to the appropriate server. In addition, processing such as data reduction can be performed as part of the event distribution process. This allows this function to execute close to where the events are detected, reducing overall network traffic.

2.4 Non-functional requirements

Non-functional requirements are concerned not with what the event processing application or platform has to do, but rather with describing *how well* it needs to do it. These requirements apply both to the event processing and to the event distribution aspects of an event processing application. They can be grouped into the following areas:

2.4.1 Performance

Performance requirements can relate to an entire flow through an event processing application, or just to a particular part of it.

Response time measures the timeliness, or latency, of the event processing and distribution. You might be concerned about the time taken between detecting an event and displaying it on a dashboard, or the time taken to detect a particular pattern. Response time requirements can be expressed in terms of:

- An upper bound of acceptable response time
- The average desirable response time

Predictability: A variance of other measure of predictability—in some applications, predictability and consistency of the latency for all events is more important than the overall speed, if a high average response time would mean that some response are very fast and some are slow

Scheduling of work and delivery of results: Multiple consumers using an event processing application can be in contention with one another. The system many need to assign priority to its workloads, to ensure that critical processing is not delayed by less important work. In some cases, such as trading applications, fair treatment of a large group of consumers, meaning that no one consumer gets priority over others, is critical.

Throughput: (often expressed as events per second) specifies the rate of incoming events that the application must be able to handle, without compromising response time or other requirements.

Scalability and Elasticity: Scalability is the ability of a system to handle growing workloads in a graceful manner, possibly with the provision of additional hardware or other resources. Elasticity is the ability of a system to scale up or down without requiring application redesign or significant operator intervention. Event processing workloads include:

- The number of event producers and consumers;
- The complexity of the processing to be performed—such as the number of operations, and the frequency at which they execute;
- The number of output events (actions) that are produced (Consider a system that is performing pattern detection (if the patterns being looked for occur frequently in the input events, then the system workload will be higher than in a system in which many of the input events can be quickly filtered out;
- The input event throughput and the complexity (including size) of the input events;
- The amount of event and other data that needs to be stored

2.4.2 Availability and Recoverability

Event processing applications typically run for extended periods of time, so the ability to sustain long term work loads is often important. This includes:

Tolerance of hardware and software faults and network failure

Ability to recover the system after a failure or major disaster

Continuous operation: the ability to effect a planned change to the application while it is still running

2.4.3 Consistency and Integrity in a distributed system

Performance and scalability requirements, and the simple fact that event producers and consumers are often located in different places, mean that most event processing applications involve multiple servers and multiple threads of execution within each server.

Many of the operations used in event processing are influenced by the time at which events occur, or by the relative order in which they occur. Implementing these operations in a distributed system involves some challenges. First, we must define what we mean by simultaneity. Second, we must address the issue of communication delays (including network breakages) when events are detected by different servers. Third, we must address the non-determinism that is introduced by the use of multiple parallel processing threads.

Different applications have different requirement in these areas, including the following:

Temporal granularity: This is the precision to be used for timestamps and time comparisons.

Repeatability: This requirement addresses the issue—if two copies of the event processing application were to be running side by side against the same set of input events, the same set of outputs should be generated in both cases.

Consistency: When multiple event consumers exist, is there a requirement that the events sent to one consumer should be consistent with those sent to the other.

Relevance and Quality of the output events: The usefulness of an event processing application depends on the quality of the events that it produces. This can be influenced by many factors, including the following:

- The precision and accuracy of the input events
- The event processing logic used by the application, particularly when the application is performing prediction
- The availability, consistency and integrity issues mentioned above
- The location at which the application is being used to detect or predict anomalous situations is an important metric is the rate of false positives or false negatives. In addition, some applications provide a probability to the events that they derive.

2.4.4 Security and privacy

Event processing systems are frequently used to make decisions that have a tangible effect on a business or on a wider community, thus, protecting them against cyber-terrorists or other subversive attacks is crucial. This process involves:

Controlling event producers access so that only authorized parties can be producers of events Checking the events submitted by these authorized parties to make sure that they are only introducing events to which they are entitled to do

Controlling application logic access to ensure than only authorized users can specify event processing logic and configure the system

These systems sometimes have to handle confidential data, including sensitive personal information, and in some cases, the application is used to derive further

information about individuals. This involves a further set of requirements, including the following:

Controlling consumers access so that only authorized parties can be consumers of events

Anonymization: Providing the ability to anonymize or remove sensitive data fields

Auditing: Providing auditable logs of the processing performed and data transmitted to third parties

In addition, the standard requirements expected of secure computing systems also apply, including the following:

Resistance to tampering: Ensuring that communication links are resistant to tampering and providing confidentiality where needed

Data protection: Ensuring data persisted to storage media is protected

Code protection: Ensuring that the system code and any application logic are protected from unauthorized access

2.4.5 Usability/Maintainability/Manageability

Many important considerations fall under this heading. First, requirements on the languages and tools must be used to define the event processing logic, as follows:

Usability: The authoring tools must appropriate to the kinds of user who will be defining or maintaining the logic, be they IT professionals, engineers, medical staff, business managers, the general public, or anyone else.

Versatility: The authoring tools must allow contributions from multiple stakeholders with different backgrounds.

Expressiveness: The language needs to scale to allow complex applications to be developed and for these applications to be maintainable, including the tools used to help check the correctness of applications.

Maintainability: In addition, event processing platforms must be manageable from an operational perspective, particularly as many event processing applications run for an extended period of time. Considerations include: Ease of adding new producers, consumers, processing logic or servers; Ease of managing the security aspects of the system; Monitoring the running of the system itself, and the event producers that are feeding events to it; Tools to help diagnose and fix problems; and disaster recovery processes and tools.

Tradeoffs must often be made among the objectives listed in this section. In some cases, a tradeoff of these objectives against the function requested of the application must be made. Adaptive load-shedding is an approach that balances these objectives. In this approach, an event processing platform stops performing less important event processing operations to ensure that it can meet the performance or other requirements of more important applications or users.

2.5 Modeling event processing applications

Modeling event processing applications is an important part of the application's lifecycle. Figure 2.1 [Etzion 2010] shows an example of such a modeling system.

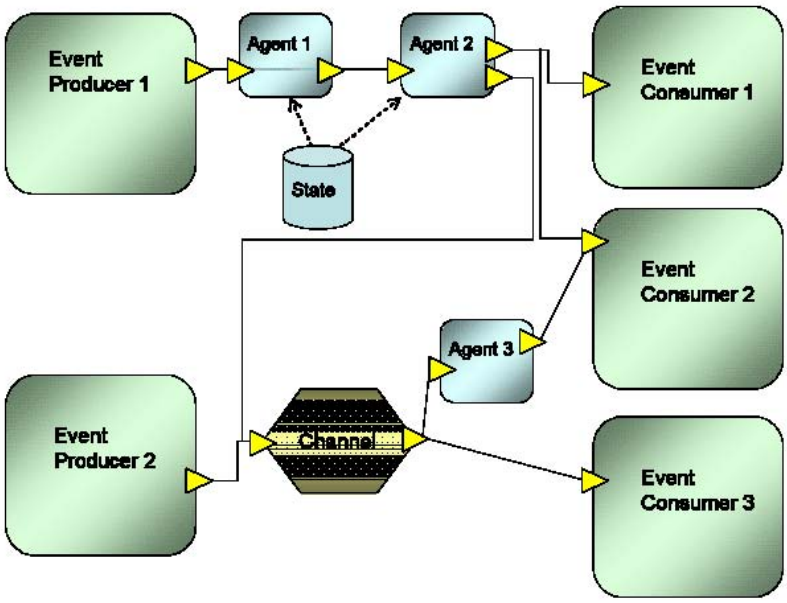


Figure 2.1. An event processing modeling for an event processing network

An event processing application model should specify the following:

The types of events used by the application and semantic metadata associated with them

The producers and consumers of events, and their relation to the event processing logic that makes up the application (In figure 2.1, this is shown as an event processing network, made up of event processing agents, representing the operations performed on events by event processing logic. Producers and consumers are linked via channels showing the flow of events.)

Context definitions (Many event processing operations depend on the context in which the event occurred, as mentioned in Section 2.3.)

Stateful data (This models data that is read or written by event processing logic as it handles events that flow through the event processing network Stateful data can include the following:

1. Historical event logs and stores,
2. State of entities external to the event processing application,
3. Shared processing state used by the application itself,
4. Reference data used to validate and enrich events.

Preferences (The preceding items all model functional requirements of the application. In addition, the model should record the critical non-functional requirements that we listed in section 2.4.)

More information about the requirements for event processing systems can be found in [Chandy 2009] and [Etzion 2010].

Chapter 3: Synergies and relations to other areas

Event processing systems and infrastructure do not exist in a vacuum, but rather entail many interactions with other technologies and computing paradigms. When considering the appropriateness of event processing for a given problem, understanding the salient differences between event processing and alternative technologies is helpful. The best approach to a problem may require blending event processing with supporting or complementary technologies. Understanding the differences and overlapping areas between these technologies and event processing is particularly useful in these cases.

3.1 An event-driven theft detection as an example of synergies and relations to other areas

The development of an event-driven sense and respond infrastructure can have an enormous impact on responses to everyday events in society.

Consider, for example, the occurrence of a theft of merchandise in a store. Loss of merchandise (known euphemistically as shrinkage in the retail industry[1]) accounts for a significant loss of revenue—typically anywhere from 1 to 8% of gross sales—and results in increased prices passed along to the public. Envision a scenario in which a person set on illicitly acquiring some merchandise enters a store. A video facial recognition system scans the person's face and processes the image to compute a set of biometric markers. The set of markers are contributed to a continuous stream that is scanned by a law enforcement application in turn linked to a police record data base.

If a positive identification is made, meaning that the person has a record, an alert will be generated to subscribers that a potential thief has been spotted in the store. In the store itself, security personnel are then alerted as they subscribe to the law enforcement monitoring system, and the store video system is instructed to provide a stream of data relating to the thief's whereabouts in the store. The detection of an actual theft may be performed by humans, although technology is currently being developed to detect gestures that might connote the action of theft and thus trigger a theft alert automatically[2]. The identification and location of the thief in the store are then sent to law enforcement along with geo-spatial information locating the store. These events are combined with real time locations of available police resources. The closest police team is alerted to respond to the theft. This scenario could be developed using a range of different technologies, some involving event processing, others independent of it.

3.2 Synergies and relations of event processing and analytics

In this section, we position event processing alongside analytics to provide guidance and a possible direction for opportunities that emerge when these disciplines are combined. Researching and developing one of the two fields with the other technology in mind will help expand the potential of such opportunities.

---

[1] http://retail.about.com/od/glossary/g/shrinkage.htm

[2] See for example http://en.wikipedia.org/wiki/Gesture_recognition and references therein.

When we refer to analytics, we foremost relate to predictive analytics in which data mining and machine learning techniques are used to sift through data for finding useful patterns and discovering new insights. The resulting models can be deployed to leverage these insights and make predictions. Common types of predictive models include classification models such as regressions, decision trees and neural nets, clustering models, and association models.

We also refer to statistical information of the execution of a system, such as measuring key performance indicators (KPIs) and business intelligence.

3.2.1 Learn and discover event processing patterns with predictive analytics

Discovering EP patterns with predictive analytics is a design-time, offline interaction in which sifting through application data can reveal insightful patterns for event processing. These patterns may benefit the system that includes event processing. Since event processing patterns are usually specified by humans, in some cases, important patterns may be overlooked or current patterns may not reach their potential (in avoiding risks or capitalizing on opportunities).

The purpose of predictive analysis is to discover and learn new patterns that can be deployed to the event processing system or engine.

In our theft example, learning and discovery tools could be used to correlate loss events with other activities in the store, such as a shift change or an especially busy time. This discovery could be used to improve theft detection. As seen in figure 3.1, the data that predictive analytics is applied to may or should include data on raw (incoming) events, events produced by event processing (derived events), and scoring model results. The models produced may turn out to be event processing rules or patterns.

Figure 3.1 illustrates the interactions between event processing and business analytics.

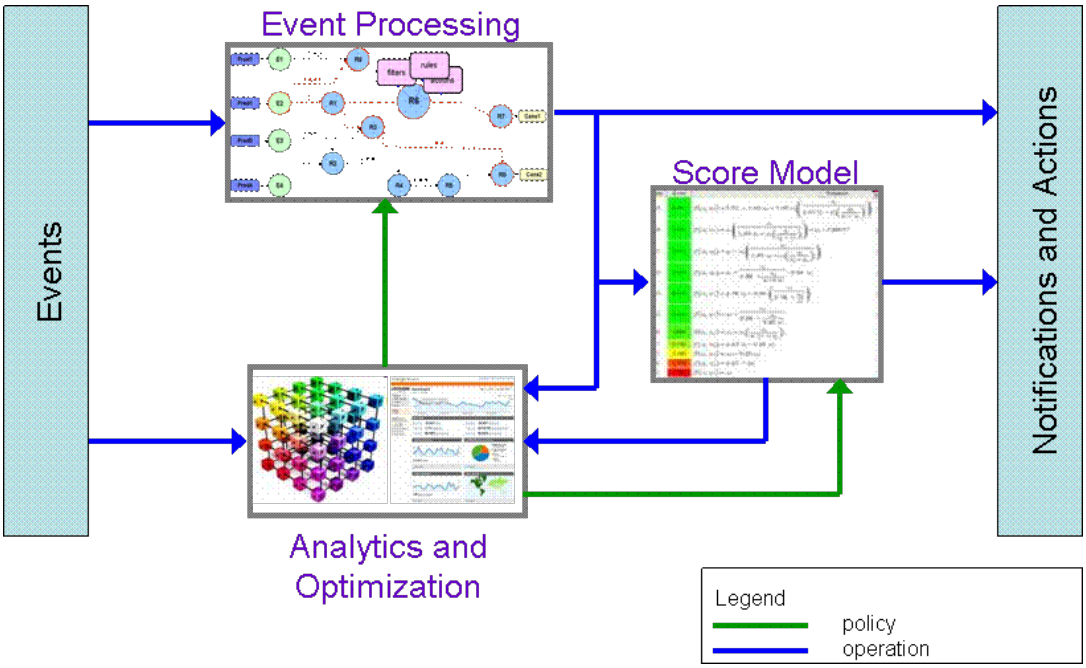


Figure 3.1: Event processing and predictive analytics interactions

3.2.2 Implementing and executing predictive models with event processing

The result of techniques that learn and discover patterns provide predictive models, but these models are often not directly deployed and executed. Sometimes these models are converted to procedural code or to SQL queries running directly on the data. At other times, the models are run offline, completely off the operational process that handles the data on which the models are to predict. Event processing may be used to execute some parts of predictive models inline with the operational process. Event processing could be used to collect, aggregate, and correlate events to provide the more informative information required to execute some predictive models. This is sometimes referred to as calculating the features' values for executing a model.

In our loss prevention example, predictive models might be integrated into police dispatch management. Crime predictions would be integrated with geospatial information to direct police resources to the best locations based on historical crime profiles and current events.

3.2.3 Incorporate predictive models in event processing

In some cases predictive models do get deployed to and executed by an engine. In these scenarios, an event processing engine/system may call upon services of that predictive model engine, such as a scoring service, and establish more sophisticated pattern detections and actions. The P-ToPSS approach [Muthusamy 2010] is one example of an event processing approach that predicts future event matches based on observed event histories and on partial event matches.

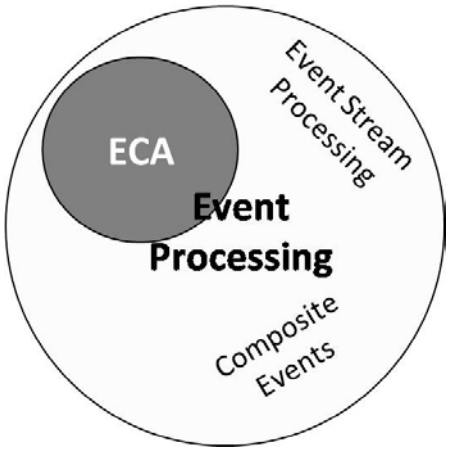
A scoring service analytic model might be used to measure the likelihood that a series of activities by a patron represent a potential theft event. If the score crosses a certain threshold, the patron could be confronted by staff to reduce the potential for a loss event.

3.3 Synergies and relations of event processing and active rules (ECA)

A strong connection exists between the event-condition-action (ECA) paradigm and event processing (EP).

ECA has its origins in active databases. The idea is as follows: If an *event* occurs that satisfies a *condition*, then an *action* takes place. Events, conditions, and actions can be more or less elaborate, depending on the system. Many systems rely on this paradigm. We can cite HiPAC [Dayal 1996] among the pioneers of this paradigm.

EP applies the three basic ECA concepts. The basic concept behind an event is the same, as are the ones behind conditions and actions. However, EP goes beyond ECA in the sense that it considers more complex events, conditions, and actions. This implies that ECA is included in EP, as illustrated in Figure 3.2.



To illustrate that ECA is fully included in EP,

Figure 3.2. Relation between event processing and business rules (ECA)

we must consider a case of events that is probably the most representative. In ECA, events were considered as simple events, in the manner that new entries in a database are typically handled. In EP, the event part tends to be more sophisticated. For instance, it may consider many events in complex expressions (e.g., in a temporal conjunction). Other extensions came over time—for instance, the inclusion of events from external sources. The fact that events are more complex leads to an E*CA paradigm. E* can then be a pattern of events, leading to the *Pattern-Condition-Action* (PCA) concept, which recently became a common term in the event literature.

In our loss prevention example, ECA rules are sufficient for some processing, such as alerting law enforcement when a theft occurs. Yet ECA rules are insufficient for more complex conditions and actions, such as identifying the best available police resource.

3.4 Synergies and relations of event processing and publish-subscribe approach

The publish-subscribe approach is concerned with the communication (dissemination) of events. The approach [Eugster 2003, Fabret 2001] is emerging as an appropriate communication paradigm for large-scale systems. It allows loose coupling between mutually anonymous components and supports many-to-many communication. In the publish/subscribe paradigm, a principal takes the role of a publisher and/or a subscriber. Principals connect to the publish/subscribe middleware in order to communicate. Publishers advertise the events they are prepared to publish. Subscribers register their interest in receiving events through a subscription that the middleware handles. Publishers produce events without any dependence on subscribers.

This process occurs through an event broker, which routes—typically in cooperation with other brokers—events from publishers to subscribers. An event is delivered to a subscriber if it matches a subscription. This process is termed notification [Eugster 2003, Fabret 2001].

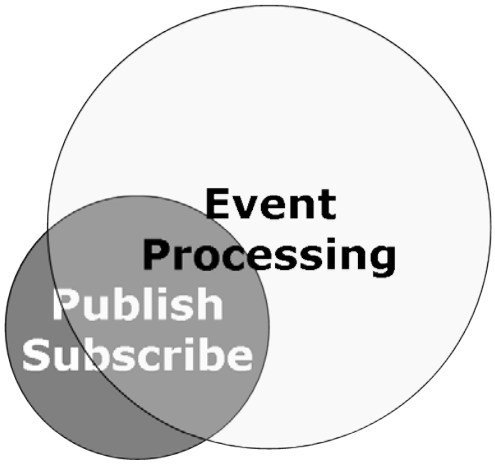


Figure 3.3. Relation between event processing and the publish-subscribe approach

Publish-subscribe systems are available as type/topic or content/attribute-based. Topic-based publish-subscribe systems involve the association of an event channel with a particular named topic/type. Producers publish events to the appropriate channel, while subscribers express their interest in receiving messages of a certain type. Content-based publish-subscribe systems consider message content—a subscriber defines his interest in receiving particular events based on the type and attribute values of the event.

3.4.1 Integrating publish-subscribe with event processing

The publish-subscribe approach is defined in terms of communicating individual events that are published and match subscriptions. The detection of patterns involving a number of different event types, also known as complex event processing, is typically built as one or more services above the dissemination network or is directly integrated into the publish-subscribe substrate as in the PADRES system [Fidler 2005, Li 2008, and Jacobsen 2009].

These services use publish-subscribe middleware. At the lowest level, they may subscribe to what is referred to as primitive events, such as sensor readings, and having detected a defined pattern, publish a higher-level composite event. In general, an event processing service may subscribe to a combination of primitive and composite events. The EP services may be deployed to distribute sub-trees of a required pattern throughout a network to minimize communication overhead. An exception to the layered model just described is the PADRES system [Jacobsen 2009], in which event processing is integrated within the broker network.

In our theft example, a publish-subscribe system might be used by the police department to distribute alerts about criminal activity, or by merchants to distribute reports amongst their peers. Without a publish-subscribe system, creating an event processing system that crosses administrative and organizational boundaries would be difficult.

3.5 Synergies and relations of event processing and business process management (BPM)

Business process management is a broad field that covers several areas, including the capturing, analysis, and design of business processes (such as the order-to-cash process). The result of these activities is often referred to as a business process model or business process type. Business process management also incorporates the automatic execution of processes according to these business process models, resulting in business process instances (e.g., fulfillment of a specific order). Process steps, as defined in the process model, can either be automated activities (e.g., calling an appropriate service) or human tasks. The steps are scheduled according to a definition of the work flow in the business process model. Finally, business process management also comprises the monitoring of business process instances, the capturing and analyzing process and step duration, and the waiting times, etc. From these monitoring activities, often referred to as business activity monitoring (BAM), bottlenecks and optimization potential can be identified.

3.5.1 Implementing BPM with event processing

Business process management can benefit from event processing in several areas. First, event processing can be used to implement the execution of business process instances [Li 2010, Muthusamy 2009]. Each step can be seen as a separate unit of execution that when finished emits an event that is then captured by an event processing engine. This in turn starts the next activity, based on the business process models. Business process models typically include rules to determine the sequence of steps that have to be evaluated by the event processing engine (complex event processing) used to determine which step is to be executed next (orchestration, choreography). Analogously, business processes can be hierarchically-layered, i.e., a process execution triggers the execution of a subordinate process.

3.5.2 Monitoring processes with event processing

Process execution monitoring is another area in which event processing is beneficial. Events can occur on the meta level (in which step x has been started) or on the data level (e.g., order received for part 124). Complex event processing is used to identify trends (e.g., execution time is growing in the afternoon) and critical situations and to throw alarms (e.g., critical order not processed within 24 hours), etc. The results can be used for real-time reporting and dashboards. The time awareness and correlation capabilities of complex event processing add functionality and flexibility to BAM.

3.5.3 Influencing business processes with event processing

In a more advanced scenario, the execution of business process instances can be influenced by event processing beyond the definitions in the process model. For example, event processing can initiate the execution of an additional activity steps in a process (e.g., due to recently detected fraud activities, all currently processed orders exceeding $100 must be reviewed by a manager). In addition, business process instances could be initiated by event processing (e.g., order material because events indicated upcoming stock shortage) or even stopped (e.g., cancel all orders from a fraudulent customer). In cases like these, complex event processing allows for flexible and timely reactions in threat or opportunity situations.

In summary, business process management can benefit from event processing, because event processing allows for continuous timely reactions and responsiveness.

Event processing adds agility, flexibility, and adaptivity to BPM, by opening up the back box of a business process. Moreover, the loose coupling that is implicit in event processing enables adding these feature in a non-intrusive manner.

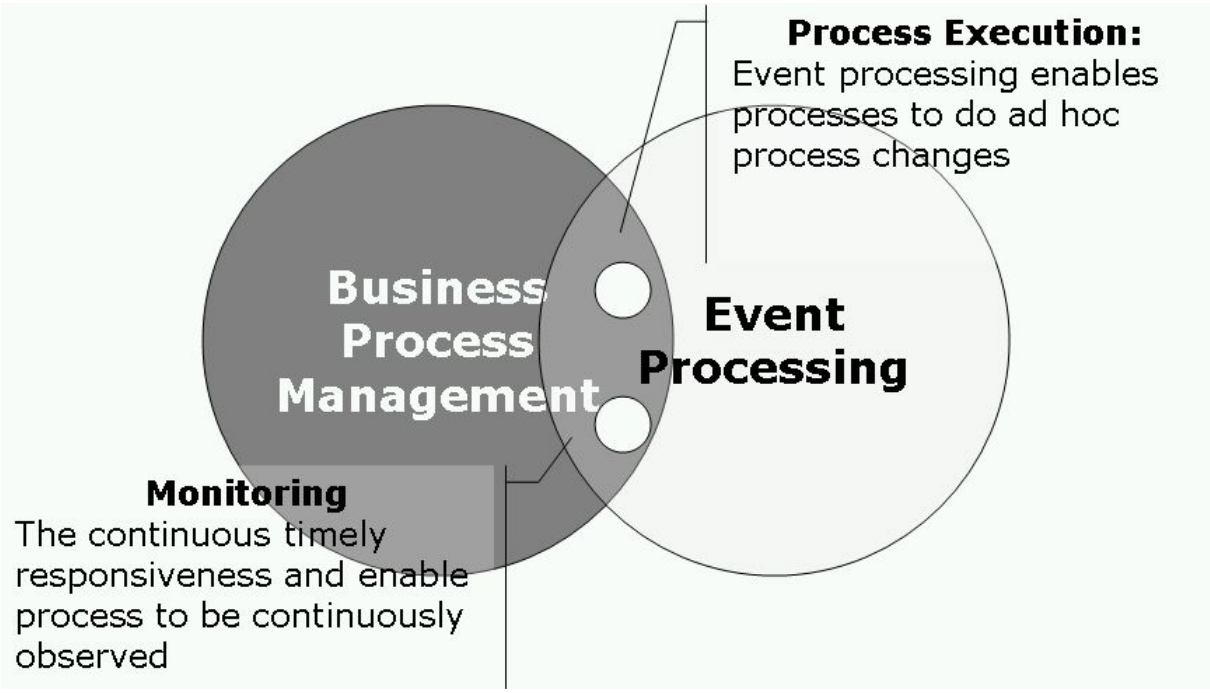


Figure 3.4. Relation between event processing and business process management (BPM)

3.6 Synergies and relationships of event processing and data streams

This section discusses the relationship between the notion of event processing and the notion of streams, an issue that is a source of frequent confusion.

3.6.1 Brief overview of streams

A data stream management system (also called a streaming system) is a platform for developing and deploying streaming applications that run continuous queries (CQs) over incoming streams. A stream is a potentially unbounded sequence of events (also called tuples), usually arranged in some order. Streaming systems typically consider an event to be a notification that contains a payload with a well-defined schema similar to table schemas in traditional databases. In some streaming systems, events also have a control field, providing metadata, such as temporal or sequencing information. Streaming systems are typically characterized by their well-defined query semantics [Barga 2007, Soule 2010] and sometimes have a declarative language [Jain 2008, LINQ]. At the execution level, queries in streaming systems consist typically of a directed graph of operators (similar to relational operators in database systems) that process events using a dataflow-style event processing paradigm. Common operators include selection, projection, join, aggregation, and grouping. A data stream is just one type of stream, and the term stream has wider coverage that includes video streams and voice streams, etc.

3.6.2 Relationship of Streams to Event Processing

The fields of stream and event processing are contemporaries, and converge in their definitions. Hence, the differences between the fields are blurry, and our observations below are merely general trends, with exceptions in many cases. We stress that the differences are less important than the similarities, which include the ability to efficiently process long-running continuous queries over sequences of events.

Complex pattern matching is generally not central to streams. Stream queries tend to be compositions of database-style operators (e.g., joins) and user-defined operators. These queries mostly represent functionality of aggregation and transformation.

Streams tend to place a higher emphasis on high data volumes with relatively fewer queries. On the other hand, event processing tends to consider the effect of sharing across many queries or many patterns as a central problem.

Video stream processing, as well as audio or multimedia stream processing, are typically not considered as scenarios for event processing, even though they are examples of a kind of stream processing.

Streaming systems, having originated mainly from the database community, tend to have a schema that is compatible with relational database schemata, whereas event processing systems support a larger variety of schema types.

In our loss prevention example, stream processing might be used to manage large volumes of data from a sensor network in a retail store, while event processing is more appropriate for the parts of the application that must generate responses.

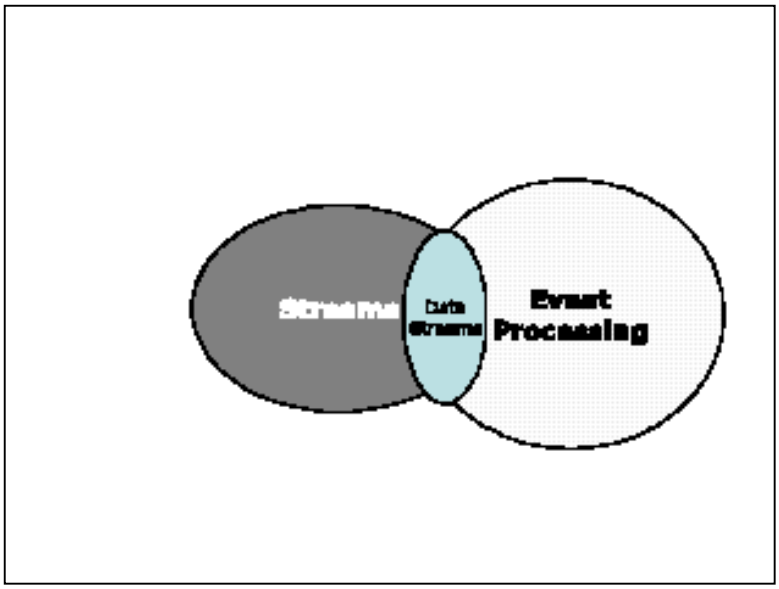


Figure 3.4 Relationship between event processing and streams

3.7 Event processing and business rules management systems (BRMS)

Due to the fact that some of the event processing models are using the term rules, event processing is sometimes confused with business rules; however, these technologies are complementary, with little overhead.

The main differences between the two approaches are as follows:

Event processing functions are activated as a direct or indirect result of event occurrence; business rules are activated on request.
Event processing functionality (as addressed in Chapter 2) typically includes filtering, transformation, aggregation and pattern detection in various forms. This overlaps with business rules, which may also do filtering and transformation, but these functions are typically stateless and do not perform aggregations or pattern detection.
Event processing has a strong relationship to temporal capabilities, i.e., temporal contexts like various types of time windows and temporal patterns. Business rules typically do not support temporal aspects.
Synergies between these technologies are typically in the form of business rules serving to make action decisions in the consumer side of event processing or business rules publishing events as part of their processing. The relationship is illustrated in Figure 3.5. Notably, extending event processing to include business rules is possible, and vice versa.

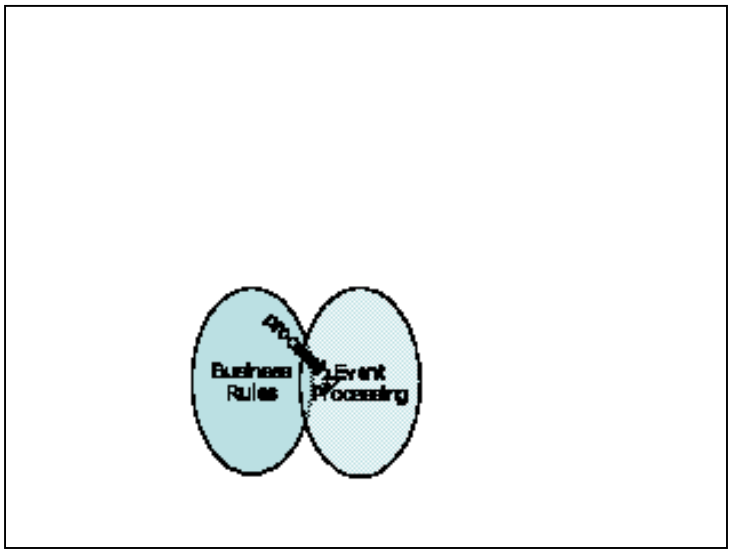


Figure 3.5 Relationship between business rules and event processing

3.8 Summary

This chapter summarized the relationship between event processing and other technologies. A specific application often has parts that fit event processing and other parts that are better suited to other technologies. Thus, achieving easy support of hybrid applications is an important challenge for the event processing area.

Chapter 4: Event processing related standards

Standards are used throughout business and IT for many reasons, including achieving a common understanding, enabling good communication, and promoting an easier interchange across organizations and tools.

Such achievements in turn lead to reduced training, a higher-quality and safer productivity, reduced costs, and increased customer confidence.

In event processing, many areas of software development reuse existing technologies and methodologies, allowing their relevant standards to be used as well. New standards may need to be developed to replace or augment existing standards.

4.1 The EP standards reference model

To understand the positioning of existing and required standards in event processing, we must understand the multiple viewpoints and business and technology areas to which standards may be applied. An EP standards reference model (ESRM) can be used to assist with this understanding. The model in Figure 4.1 uses the OMG model-driven architecture (MDA) viewpoint to describe standardization areas.

4.1.1 Business and technical perspectives

The ESRM describes a set of components influencing the development, implementation, and deployment of EP applications, whereas EP refers to general event processing. The model extends the MDA, in which a sequence of models are defined from a business or computer independent model (CIM), and then progressively detailed or transformed with IT perspectives, first to the platform independent model (PIM) and then to the platform specific model (PSM). These models explicitly distinguish between the business and the technical perspectives.

On the one hand, the aim of the business perspective is to define the domain terminology independent of any specific technology in a specification sheet. This provides business users and managers or decision makers, and for IT systems, the developers, with a common understanding regarding the general definitions of a system from which requirements can be abstracted. These models are often augmented by business use cases. The abstraction from business model to IT system requires answers to questions such as: What functionality must be realized? What limitations must be considered? What best practices should be considered?

On the other hand, the technical perspective aims to describe, in increasing detail, the implementation techniques and technologies for developing some automated aspect of the business perspective. The target groups here are the software architects, designers, and developers. The main questions here are: What technologies have to be used? What technical guidelines and patterns should be considered? What technical conditions have to be considered?

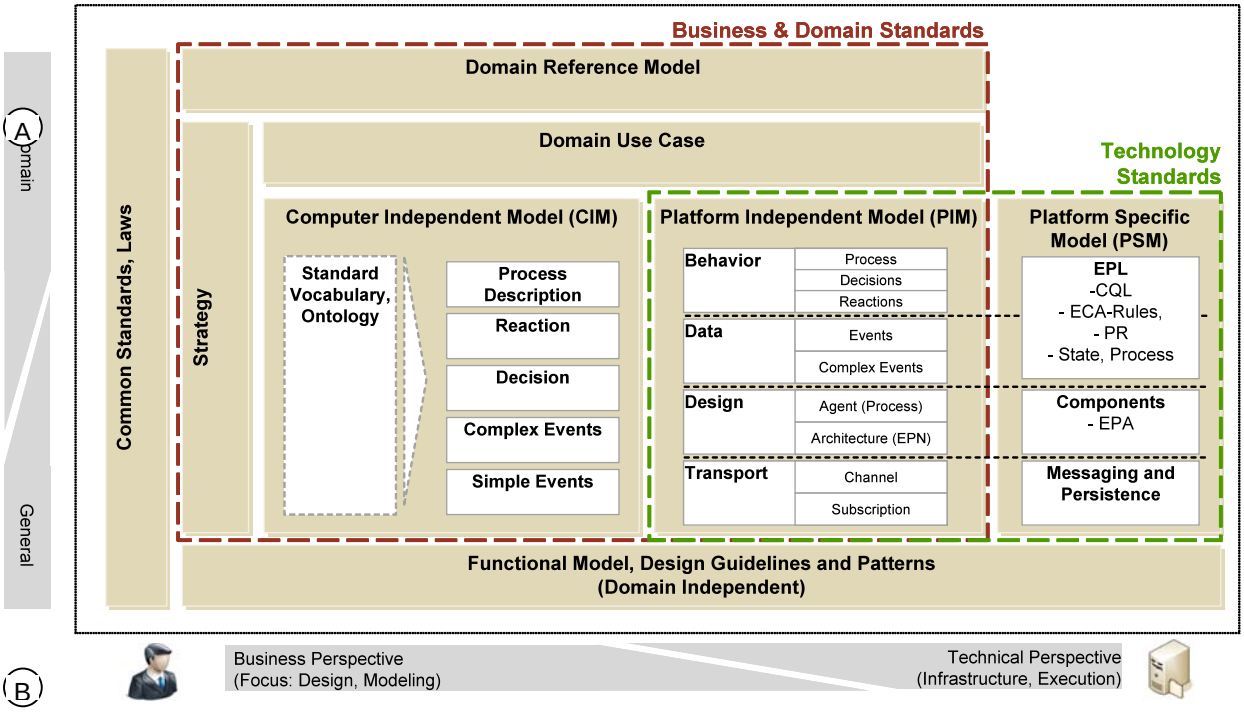


Figure 4.1 EP standards reference model—ESRM

4.1.2 Domain-specific and general standards

The ESRM also illustrates that both domain standards and general standards support EP.

Domain standards are specific to a business domain, e.g., to banking, retail, or logistics. Many domains have their own specifications and regulations that are adopted for IT implementations and also are relevant in EP applications. For example, a domain standard called ACORD Framework provides an XML model of the concepts used in insurance underwriting activities and is applied to certain classes of insurance. This standard can be adopted in EP as a data model or extended to provide an event model in EP systems.

General standards consist of domain-independent standards, e.g., common adapted design guidelines and patterns in the field of software development. The UML standard, for example, provides a framework of modeling components, such as class diagrams and state models that are relevant to EP systems.

4.2 Standards per the ESRM classification

The following sections provide a short introduction to the CSRM component standards that influence EP applications.

4.2.1 Common standards and laws

Common standards and laws provide conditions and regulations that have to be considered in business, such as laws for stock trading. Development of a software application must follow these laws.

4.2.2 Domain reference model

Domain reference models provide a subset of best practices, reference business processes, etc., for a specific business domain. These models help adaptors in defining a suitable application, based on well-proven best practices, such as a well-proven reference business process for the manufacturing domain.

4.2.3 Domain use case

The use case describes how a problem can be solved, e.g., how to detect fraud in stock trading. Use cases are often derived from the business strategy and the reference model.

4.2.4 Strategy

The strategy defines the individual business strategy that has to be considered in developing new applications for a company. The strategy consists of business (business motivations, goals, policies, and rules) as well of technical (applications infrastructure and allowed frameworks) conditions.

4.2.5 Functional model

The functional model provides domain-independent best practices and guidelines helping to design and develop a proper application, e.g., “Design Patterns—Elements of Reusable Object-Oriented Software”.

4.2.6 Computer-independent model

The computer-independent model (CIM) describes a business functionality or system without reference to IT specifics. CIM should provide a common understanding between business users and software developers. In the EP model, CIM consists of following components:

Standard vocabulary or ontology: The ontology provides a set of definitions and terms enabling a common understanding for all stakeholders. Much like a glossary, the aim of an ontology is that every participant has to use the same word for a defined concept.
Process description: The process description details the activities and flows that produce a specific product or service, enhanced with events occurring in it or influencing it.
Reaction: The reaction is a definition of the activities that must be initiated, based on previously made decisions.
Decision: The decision refers to a definition of rules, that is, what has to be done if a relevant situation was detected by pattern matching.
Patterns: Patterns refer to the definition of event patterns for detection of relevant situations that represent knowledge from source events, e.g., for fraud detection.

Event schema: This refers to the definition of attributes and types consisting of a simple event, e.g., the event “Stock Price” comprises several fields, such as security identification code and amount, etc.

4.3 Platform independent model standards (PIM)

The platform independent model layer represents behavior, data, design, and messaging, independent from a particular EP platform. These PIM abstractions support increased portability and platform independence and cross-platform interoperability and interchange between domain boundaries, as described below.

Event-driven behavior: Effects of events lead to some (state) changes in the properties of the world, which can be abstracted into situations. Decisions represent the choices a system can make in certain situations. Actions might be triggered or performed as reactions, based on the decisions and changes in states or situations as an effect of events.

Event Data: Platform-independent representation of events and their data is crucial for the interoperation between EP tools and domain boundaries. This interoperation can refer to any of the following:

1. Interoperation among different EP products to exploit benefits, e.g. stream + rule processing component
2. Interchange of events in a distributed heterogeneous EPN
3. Interchange of events over domain boundaries, e.g., cross-organizational processes

Event processing design: Platform-independent (reference) architecture and design models addressing different views for different stakeholders can achieve the following:

1. Furnishing abstractions and reference generalizations to manage technical complexity
2. Providing structure for solving design problems
3. Experimenting to explore multiple solutions, including best practice solutions such as design patterns, reference architecture descriptions that model the abstract architectural design elements, and architectural reference models that describe the important concepts and relationships in the design space

Messaging: PIM messaging addresses transport protocols and routing as well as coordination/negotiation mechanisms.

4.4 Standards in ESRM areas:

Table 4.1 describes the standards in the ESRM areas.

Area	Available Standards	Benefits	Gap	Action
common standards, laws	several laws (often country specific), e.g., BörsG for stock trading in Germany	no benefit for EP	none	none, because not EP-specific
domain reference model	various per domain for data	efficient design and development of EP applications for specific domains	rarely handles events, just data	extends to common events as required
domain use case	covered by - UML use case - EPTS use case template	creates common understanding between business user and software developer	misses event aspect	improves use case templates
strategy	Usually a textual description; partial coverage in - OMG BMM	clarifies business strategy	may need more emphasis on events	none, because not EP-specific
functional model	- EPTS Fn Ref Architecture - many different ones in the field of software development	helps implementing a proper application	specific functional patterns for EP not available	creates and improves functional models for EP
standard vocabulary, ontology	- text based glossary - KR ontologies (e.g. in OWL) - OMG SBVR	common understanding for all stakeholders involved	time ontology. little emphasis on events in such vocabularies.	none, because not EP-specific
process description	- BPMN - EPC	creates a common understanding between business user and software developer on a “big picture”	insufficient detail on events for EP applications	extends BPMN with sufficient support for modeling simple and complex events
reaction	can be an event update through to a service definition	common understanding for all stakeholders involved	none	none, text-based description is sufficient
decision	none but - decision table, tree, etc (OMG DMN proposed)	common understanding for all stakeholders involved	none	none, because not EP-specific
event patterns and other event building blocks	none, but - OMG EMP proposed	better understanding of relationship between involved events and roles	no structured way to describe EP building blocks, such as: patterns, aggregations, transformations, etc.	new standard required
event schema	- UML - design language used in NEAR [Ammon 2009]	creates a common understanding across business users/event sources and developers	not sufficient for needs of event processing	improve modeling languages, e.g., NEAR [Ammon 2009]

Area	Available Standards	Benefits	Gap	Action
event-driven behavior	- W3C RIF (and RuleML) - OMG PRR - OMG UML behavioral view diagrams - OMG BPEL, W3C WS choreography [OMG EDA]	declarative, explicit representation of behavioral/reactive logic, publication, and interchange of decisions and reactive behavior	standards for specific domains: rule-based event processing languages (RuleML), Web service execution (BPEL, WS-C,…); SQL extensions [OMG EDA]	- rules: further standardization in W3C RIF [RuleML] - standards for other domains needed, e.g. stream processing (SQL extensions) [Jain 2009] - interoperation, e.g., rule standards with BPEL
event data	- software engineering: UML structural view diagrams - knowledge representation: many event ontologies exist (e.g. in OWL - rules: W3C RIF/RuleML - OASIS WS notification, W3C WS eventing, OASIS WS Topics - OMG event meta model - OASIS common base event	- declarative representation, translation and interchange of events - interoperation between different platform specific tools and domain boundaries (requires semantics)	standards for specific domains: rule-based event processing languages (RuleML), Web Service Events (WS X), enterprise applications (OASIS CBE)	- rules: further standardization in W3CRIF/RuleML - standards for other domains needed, e.g., stream processing - interoperation, e.g., rule standards with other event standards/ontologies
event processing design	- UML 2 implementation view diagrams - ISO/IEC 42010:2007 recommended practice for architectural description of software-intensive systems - agents: OMG agent metamodel; FIPA agent model; … - workflow management coalition reference model	- abstraction from the PSM design increases understandability and reusability - agent model is an abstraction from technical IT components to role-based agents on a more abstract level	current standards are not specialized for event processing design	reuse and extend existing standards for event processing design descriptions
messaging	- many transport protocols: JMS, JDBC, TCP, UDP, multicast, http, servlet, SMTP, POP3, file, XMPP - message routing patterns - coordination and negotiation mechanisms/patterns [OMG EDA]	platform-independent messaging	none–existing standards can be reused for transporting and messaging events	None

Table 4.1 ESRM-related standards

4.5 Next steps (action items)

We propose the following action items for the EP community in the area of EP-related standards:

Create a set of DSRMs (with potential use cases, events, complex events, etc.) to enable visibility in companies and show the potential of event focus and EP as well as providing guidance on the implementation of EP applications

Create computational-independent common notations for modeling EP specific things (e.g., syntactic and semantic extension of UML class diagram for modeling event patterns and other event processing building blocks and their semantics (event ontology), mappings into other representation formats (e.g. W3C OWL event ontologies), and extension of BPMN with EP, etc.)

Extend or introduce platform-independent technical standards to increase interoperability and reduce implementation effort when using EP in applications. Standards efforts for rule-based event processing languages already exist (such as the RuleML effort) and will be further standardized (as in W3C RIF and OMG PRR extensions). For other domains, such as event stream processing and event querying (e.g. SQL extensions), some initial standardization efforts were attempted, but a standard has yet to emerge [Jain 2009]. Another interesting issue to address is the mapping between domain standards, e.g., between rule-based event processing and event stream processing, that will enable the hybrid interoperation among different event processing technologies.

Document existing relevant standards per the EP standards reference model for guidance and consideration. (See the EPTS Wiki for more information.)

Chapter 5: Grand challenge: The global event processing fabric and its applications

This section identifies a grand challenge that serves as a common goal and mechanism for coordinating research across the spectrum of people working on event processing.

Event processing has many challenges, many of which are grand; however, our goal here is limited: identify a single, though broad challenge that impacts society. The challenge should provide a basis for measuring progress of the EP community. The grand challenge should be revisited in regular sessions at annual meetings, such as at the DEBS conferences, to review progress and discuss approaches to the challenge.

The event processing grand challenge (EPGC) comprises two parts:

A decentralized, global, Internet-like infrastructure, which we call the Event Processing Fabric, built upon widely-accepted open standards
The design, development, deployment, and management of life-changing, or society-changing applications that utilize the Event Processing Fabric
This challenge is grand in the sense that its realization requires new cutting-edge R&D results and that it will influence society in a transformational way—meaning going from being a society that merely reacts to problems, to a society that proactively exploits opportunities and guards against threats.

In the following sections, we describe the Event Processing Fabric part of the grand challenge and provide the basic building blocks for its efficient realization. We then list life- or society-changing applications that could utilize this fabric. Finally, we present some related work and give concluding remarks.

5.1 The Event Processing Fabric grand challenge

The most important part of the EPGC is to create an infrastructure, built upon widely-accepted open standards that enable different components of the event processing architecture to be plugged into an event processing "fabric" with minimal effort. The Internet grew rapidly because plugging into the network following IP protocols was easy.

Our challenge is to develop an Event Processing Fabric into which components can be easily plugged and unplugged, allowing for the development of time-driven or event-based global applications. We are considering the run-time dynamics of the whole architecture—not only should components be dynamically pluggable, but complex event patterns and their subscriptions should also be defined and discovered dynamically as well as be composable.

This "on-the-fly-adaptive" nature of the Fabric will enable the dynamic definition of the situations of interest (complex event patterns) and *ad hoc* generation of timely reactions to new situations. The key challenge is continual global situational awareness, realized in a decentralized fashion. This will enable going beyond the (early) recognition of problems toward the (proactive) discovery of opportunities and threats.

A service that senses activity related to interests and then responds appropriately is just one example that illustrates the adaptive nature to which we refer.

Suppose that you plan to buy an eco-friendly car. A service could create and continually update a dynamic report relevant to that interest. This report could include reviews of relevant cars, prices, locations of sales, identities of friends who have bought such cars recently, and locations of friends in the vicinity who own that car. All of this is already doable; the challenge is to develop a fabric facilitates plug-and-play, making doing it much easier.

The Event Processing Fabric will incorporate the precision and timely feedback of the global positioning system (GPS); the distributed ownership and reach of the World Wide Web (WWW) the community-based, self-curated, constantly-updated content of Wikipedia; and the adaptive nature of complex adaptive systems. The Fabric will be used to develop applications that range from systems as complex as detecting incoming earthquakes to systems that simply warn of schedule changes in daily commutes between work and home.

Technically, the challenge of building the Fabric consists of having thousands or millions of different sources that collect sensor data from across the globe, and then filtering, aggregating, transforming, and detecting patterns of interest in real-time and historical information. In addition, the Fabric will need to manage the subscriptions and locations of millions of users, in a secure and anonymous way, across different geographic and administrative domains, sending alerts in a timely fashion and utilizing the most appropriate channels of communication.

As in other domains, the infrastructure network used to build the Event Processing Fabric will be open and will be used by private and public agents. For example, manufacturing industries can use the Fabric to instrument production lines and issue warnings as needed. Energy providers can instrument their energy grids to monitor for safety levels and warn people and cut off systems when necessary. Airlines can inform users of changed schedules; health organizations can monitor check-in types and numbers, make predictions, and raise alarms of epidemics. Schools can alert parents on local events. In short, the Event Processing Fabric is designed to be the highway of global, real-time data, and the enabler of applications for a proactive society.

5.2 Event Processing Fabric implementation issues

In the following sections, we identify some of the implementation issues and quality attributes related to creating the Event Processing Fabric. Our Event Processing Technical Society (EPTS), with its roots on databases, rule-engines, publish-subscribe systems, distributed systems, and other systems, is especially positioned to overcome these challenges and make this vision a reality.

5.2.1 The Fabric

The Event Processing Fabric should help develop many (but not all) event processing applications. For example, the Fabric may be inappropriate for highly secure applications such as military or homeland security. Likewise, the Fabric may be unsuitable for high-performance applications such as real-time stock trading. Similar to the Internet, the Fabric will be extremely useful, but not the only way to connect components.

A critical challenge in developing a plug-and-play Fabric is agreeing to a single or a couple of model architectures for event processing. Assuming that the EPTS architecture group has determined the architecture(s) of choice, the event

processing community now has to develop standards for data modeling and schemata for communication. For example, the community may have to agree on how components register themselves with the system, and how the system monitors the health of its event processing network.

The event processing community can also exploit web services standards for communication and managing the network. Agreeing on a single standard is perhaps too much to hope for, but we can build upon the most popular ones, to the same extent that the Web services community has agreed on two or three standards.

5.2.2 Quality attributes

The Event Processing Fabric must provide quality attributes reliability-based on service level agreements (SLAs), ensuring the agreed quality of services in the following areas.

Privacy: ensuring the confidentiality of published information
Security: Protection from hackers that attack the fabric
Openness (Interoperability): plug-and-play standards are necessary for the fabric to be a fabric
Provenance: tracing back a chain of events should always be possible
Elastic performance: accommodating variable requirements, possible using so called “cloud” services (requirements for extreme-scale disasters vary dramatically over time and the requirements are likely heaviest when disasters occur)
Energy-efficiency: minimizing the energy consumption of the devices connected to it
The following attributes should be discussed:

Autonomic computing support: the Fabric must be able to self-tune itself
Intuitive User Interfaces: research needs to be done to ensure that the broad public can tailor the fabric for their own needs
Quality of the complex event pattern bases: ensuring the validity of the situations of interests
Non-repudiation: identifying *who* produced what should always be possible
Authentication: some services should force consumers or producers to identify themselves
Anonymity: and at the same time, others services might allow anonymous users
Availability: the systems should be available at all times, even in the presence of network partitions; eventual consistency is preferable to always consistency as a tradeoff for obtaining higher availability
Quality-of-Service: the fabric should allow differentiated quality-of-service levels or priorities for different service agreements.

5.2.3 Business and societal adaptation

An essential part of the application development is the definition of the revenue models on which the applications are operating, assuming that these applications will generate added (business and societal) value for all stakeholders. The development of the mechanisms for involving event producers in the model, e.g., what would motivate them to make their events available for others, is especially challenging. Similarly, event consumers must be ready to accept a selected pricing model.

Further, pattern engineering must be applied to support for the creation and maintenance of complex event patterns and must be provided in a user-friendly way.

These concepts should be built independently from the technical realization of the Fabric.

5.3 Applications utilizing the Event Processing Fabric

The second part of the Event Processing Grand Challenge is to demonstrate the use of the Event Processing Fabric for different kinds of interrelated applications, such as the following:

Extreme-scale disasters: This use of the Fabric would help society respond to situations such as hurricanes, earthquakes, and terrorist attacks. Data sources include sensors managed by government agencies (e.g., meteorological services tracking storms), companies (e.g., companies that monitor congestion in roads), and individuals (e.g., web sites that contain images or videos of situations such as forest fires and buildings after quakes).
Critical societal applications: This refers to managing systems such as the smart electric grid, the smart city, and home healthcare for the aged. These systems are becoming increasingly event-driven. For example, as the electric grid grows, it relies on green sources of energy, such as wind and solar power, which are less dependable that other energy sources; thus predicting, detecting, and responding to events, such as cloud cover over solar arrays or sudden drops in wind speed, becomes critical.
Personal applications: This refers to determining the optimum commute using buses, trains, or taxis, based on sensing resources' locations, availability, and on planning the best routes.
Social “eventing”: This is the next step in the evolution of social networking. Data sources in social networks, such as Twitter and Facebook, are proliferating. Many organizations are working on mining this data. The challenge is to use the Event Processing Fabric to complete the loop from data acquisition to information fusion, planning, and responding, so as to allow members of a social networks to monitor their friends in the network and to easily respond to their actions.

5.4 Elements of the challenge

The EPGC exercises most of the components of the event processing architecture (see the EPTS group on reference architectures or books on event processing). These components include:

Data acquisition components such as sensors or software agents that poll Web sites and human beings
Event processing agents that integrate data from multiple sources over time to estimate the states of the world relevant to the given application (These components may estimate the probabilities of different future trajectories of the state of the world and plan sequences of actions to deal these outcomes. The agents will be able to reason about the world using available domain knowledge; the event processing agents will include an extended functionality to support proactive behavior such as prediction and decision agents.)
Responders or actuators that execute actions
Communication networks for transmitting information between components (These networks may be implemented using publish-subscribe protocols)
Management components used to specify, monitor, and control the entire system (They enable design- and run-time configurability of the system and include self* functionalities such as self-evolving nature.)

5.5 Related work

Currently, the system most closely related to what we have been describing as the Event Processing Fabric is pachube.com. Pachube is a site to “store, share & discover

real-time sensor, energy and environment data from objects, devices & buildings around the world." Pachube is a convenient, secure, and scalable platform that helps you connect to and build the "internet of things".[3] However, Pachube does not support the requirements described in Section 5.1, nor the availability, partition tolerance event-driven architecture we would expect from a wider-encompassing Fabric.

Different communities at different times have proposed grand challenges, and some of these earlier proposals have many points of similarity with the challenges identified at Dagstuhl. The grand challenges identified by the database community[4], EPTS at Trento, the CANOE workshop, and elsewhere, are particularly relevant in identifying the reasons for selecting grand challenges, and for some of the challenges we have proposed. The event processing community plans to collaborate with other groups on grand challenge problems that share similar features.

5.6. Summary

We see this grand challenge as the vision for developing the event processing community in the next decade. The challenge will help to synchronize efforts at different parts of the community. Indeed, the research community will learn what the breakthrough applications are; analysts will learn how these applications will transform business and society; vendors will learn what the unique selling points of the event-driven solutions are; and customers will learn what else they can expect from the technology.

The EPTS community will plant the seeds for this work, while the challenge will be used for measuring progress in the community itself. Extracting a set of small-sized challenges out of the big one also has potential to challenge the community to realize goals in a shorter time frame (e.g., in the form of Demo challenge at the DEBS conference).

[3] http://www.pachube.com/

[4] The Lowell Self-Assessment Report," CACM, Vol. 48, #5, May 2005,

Chapter 6: Near-term research

While the identified grand challenges for event processing offer clear directions for longer-term needs, we can also state a number of more near-term goals. These are based on the gap between present needs and the identified common functionalities that are at our disposal for EP.

This chapter outlines research directions that have been recognized as necessary to fulfill the present requirements posed to EP systems. Some of the outlined directions may have already undergone initial investigations yielding preliminary results or may already be subject to vigorous research. Others may still be on the wish-list.

We coarsely regroup the research tasks that constitute our proposed research agenda into four themes.

- Event semantics include issues of identifying and extending the meaning of events, reconciling the view of real-life events and that of programming artifacts.
- Events and actions make up an overarching theme motivated by the observation that the boundaries of EP can be extended to cover many more present and future application needs by abandoning the early limited view of EP applications. The event driven architecture consists of three largely orthogonal tiers for event production, event processing, and event consumption. For instance, considering actions triggered in response to events to be integral parts of an EP application opens many new avenues.
- Systems' efforts are not bluntly geared at squeezing out the next few % in performance, but rather aim at more far-reaching improvements in performance that consider reliability and consistency guarantees offered to EP applications.
- With the scale and reach of EP applications increasing, privacy and security become prime concerns, as for any distributed applications. However, EP poses its own specific challenges in this area.

6.1 Event semantics

What are events? What do they represent? What are their types, their attributes? What can be done with them? The goal of this first axis of research is less to reconcile all existing models of events, but more to further the semantics of events along different lines. The primary goal is to extend expressivity of EP systems to accommodate new applications and more efficiently support existing ones.

6.1.1 Probabilistic events

In most contexts, a fine but clear distinction exists between actual events, reifications, and notifications of such events subsequent to observation. Since any observation can be perturbed, we can view notifications as describing events having occurred in the captured form with a given level of certainty, or with their associated data with a given confidence. Uncertainty of data is an area that has been widely explored in the context of databases. As event reifications are usually associated with data, this data would naturally seem to come with a degree of certainty or confidence as well. Such degrees can also be associated with predictions of future events, which will be addressed. At the same time, we can often associate levels of relevance with events that may or may not be correlated with their confidence levels. We refer to such models, which assign non-binary weights to events, as supporting probabilistic events. This is the case whether the weight metrics are continuous or

discrete or actually represent a probability. We foresee that the handling of such events and the reasoning about corresponding event combinations can benefit strongly from probability theory and stochastics. Models of probabilistic EP, as well as corresponding programming and system support, are necessary.

6.1.2 Provenance

Provenance is an important aspect of events that is related to the above and that has been largely under-addressed so far. Provenance is related to probabilistic events in that the origin of events is another relevant attribute that can influence their handling and processing. We thus expect that research on this topic may benefit from synergies with that of probabilistic events.

Provenance, however, also has other flavors, which can be exploited for audit and blame assignment. The fundamental theory of events in concurrent and distributed systems provides models for reasoning about event relations (e.g., causality), and mechanisms for identifying instances of such relations (e.g., logical or vector clocks). These models can benefit from research on support for data provenance, but additional research is required in the event processing context. A system can hardly track all event relations and store all events ever observed during application execution to allow any event to carry its entire genealogy. In addition, tracking actions that led to given events or that have been performed during the handling of causally preceding events might also be desirable. Research is required in programming models to allow applications to specify relevant subsets of event genealogy that need to be captured, and system support is required to effectively and efficiently obtain such traces.

6.1.3 Event context

Real-world events are most commonly associated with time and space dimensions, mirroring the most common inquiries about such events, namely *when* and *where*. However, event context can encompass more information and items than these two well-known dimensions entail, as illustrated by the previously proposed lines of research. Origin, genealogy, or weight can all be viewed as describing parts of the context of an event. In addition, they illustrate the importance of the relative and logical notions of *when* and *where*, as opposed to absolute, physical ones.

Typically, distributed systems may be devoid of synchronized clocks (motivating logical time to reason about time) and applications might be more concerned (or at least equally) with which software component or module created a given event rather than the host on which it actually happened. One way to tackle event contexts and also to support the above research directions could be to investigate all-encompassing notions of an event context capable of modeling all existing event context attributes. An alternate, perhaps better, approach is to undertake specific research into versatile support for context. This could be tailored to different needs for specific applications, infrastructures, individual components, or paths in applications. As hinted to in the context of event relations, this may be necessary to deal with the sheer complexity of context in EP applications. This approach will also better accommodate new types of contexts such as actions leading to respective events. This is further elaborated in the following sections.

6.2 Events and actions

Perhaps one of the most important near- to medium-term extensions of established EP paradigms and systems we foresee is the consideration of a larger picture that

extends beyond capturing events and event-handling activities (reactions) in applications, and delegating these to dedicated EP systems. In the enlarged perspective we envision broader consideration of actions—of which reactions are but a special case. This allows us to fully close the loop encompassing event producers and event consumers, thus providing a more holistic view of event-based applications and broadening the scope of EP.

6.2.1 Complex actions

It is important to consider actions as part of an EP application, in addition to making allowances for complex actions. Complex events paved the way for much more expressive applications as opposed to single event processing scenarios. Similarly, support is required for the expression and efficient handling of complex actions that are triggered in response to complex events. In particular, complex actions—whether or not reactions to lower-level events—can include the generation of events, closing the loop between event producers and consumers. We foresee programming models and abstractions exposing complex actions, so that static analyses can be devised to statically verify properties such as the absence of deadlocks or the satisfiability of timing constraints. To achieve completeness, these analyses may be complemented by runtime monitoring techniques. Such dynamic analyses can also be employed to optimize the scheduling of complex actions, based on the same time insights into their semantics and boundaries.

6.2.2 Goal-directed reaction

Our goal is to move away from a simple three-tier view of separate entities—(a) event producers, (b) EP system, and (c) event consumers—to a larger picture that also provides opportunities for higher-level perspectives on EP-based applications. Most importantly, a larger picture enables more declarative-style and goal-directed specification of complex applications. This gives EP the nature of an implementation technique rather than that of a programming paradigm. This approach includes models defining higher-level goals based on rules, predicates, or invariants. We foresee the use of automatic theorem in combination with the axiomatization of corresponding EP programs to prove that high-level properties are ensured by lower-level EP reaction strategies and implementations.

6.2.3 Compensation and retraction

In some cases there might be achieved by defining alternative reactions to complex events; Furthermore, strategies identified as suitable at a given point during execution may later reveal themselves to be sub-optimal or even erroneous. Often, these strategies have already been pursued and corresponding actions triggered. In such cases, compensation for the triggered actions might be necessary and the only viable solution is to reestablish invariants or to realign the system state with higher-level goals. Moreover, when actions have been triggered outside of the EP application, these actions cannot simply be undone, but can only be made up for by well-defined compensating actions. For the remaining execution, corresponding reactions or execution paths might be identified as being unsuitable or may later be re-installed as valid strategies.

Although the foundations of compensation have been explored in the context of more traditional data-centered and transactional computation models, little effort has been undertaken in combination with the more (inter-)action-centric abstraction of events. Existing work on transactional events focuses on backward (roll-back) recovery mechanisms, rather than forward recovery. Concerted advances at the programming

model and system levels are necessary to support compensation. We believe that this can lead to powerful programming models, possibly in combination with advances on event context for tracking backward to causally preceding events and actions.

6.2.4 Predictions and speculations

Predictions and *speculations* are two more advanced EP features that are of great importance for present and near-future applications. The former concept can be understood as consisting of using EP to predict future happenings based on past events, in combination with stated expectations or with complex events observed in the past. Predictions can be useful to prepare for actual actions whenever the predictions should become true, or to immediately trigger preemptive measures. Predictions can be combined with probabilistic events, as proposed in Section 6.1.1, to associate predictions with confidence intervals. With actions being viewed as an integral part of an EP application, and able to give rise to events, we can take predictions a step further towards the latter notion of speculations. The integration of an EP infrastructure with a simulation engine can allow the speculative execution of specific reactions to predicted complex events. This can further extend the horizon of prediction. If the effects of such speculative actions can be contained, they can simply be undone in cases when an antecedent prediction is invalidated. Otherwise, compensation can be useful in this context to counteract any speculative actions. Also here, advances at the programming model as well as EP system level are necessary to offer such functionality.

6.2.5 Adaptive event processing

The early rather static models of event processing consist of simple, obscure, reactions for handling simple events. Once we abandon these to embrace a more complete view of EP applications, we can easily conclude that (a) reactions to given events or complex events cannot be assumed to stay the same forever, and that (b) even the (complex) events that are of interest to an EP application or a given component of such an application may vary throughout its lifetime. Achieving such adaptive event processing will require advances on various fronts. Due to the rapid growth of the market for EP applications, many quick and pragmatic solutions were necessary to satisfy the needs of that time. In hindsight, such solutions too often consisted of adapting or even simply adopting existing components designed for related problems, many of which eventually became part of established best practices in the field. Providing more dynamic and adaptive EP solutions will thus require rethinking larger parts of the protocol stacks of EP systems. These efforts will be driven by the creative tasks of identifying and envisioning which parts of EP models must or can be replaced to allow for more adaptivity in EP applications and how this can be done.

6.3 Event processing systems

As outlined in Chapter 2, EP applications are centered around an EP infrastructure, which deals with routing, filtering, and composition of events based on event processing agents (EPAs). Even considering only current EP system functionalities, several advances are necessary and already conceivable on this front.

6.3.1 Function placement and optimization

A major key to the performance of EP applications consists of breaking down event treatment into elementary operations, and placing them over a set of physical (e.g., computers and processors) and logical (e.g., processes and threads) entities capable

of hosting EPAs. While a decent amount of effort has already been put into such decentralization in the past, we see room for improvement in dynamic placement strategies with proactive behavior, based on fluctuations in application load and load distribution. Current models and systems react to spikes in activity, at best,. Moreover, they consider event processing operations to be static and do not support instance-adaptive or speculative event processing. Last, but not least, any further additions to event semantics or work on the refinement of actions will yield new challenges and new opportunities for dynamic placement strategies, including the placement and scheduling of complex actions that are to be performed in addition to EP.

6.3.2 Consistency

Literature is abundant on pragmatic load-shedding techniques to deal with spikes in event production and in other activity from event-based applications. Despite early work on more selective mechanisms for shedding load, we observe that little work has been done on consistency and other properties, with respect to individual events or especially with respect to complex events. Typically, current decentralized approaches for event composition are pragmatic and primarily consider shortest-path routing and similar metrics for ensuring low-latency when placing EPAs, often times duplicating such EPAs. As a result, different EPAs may be seeing and combining the same events, but in different orders and arrangements. As a net effect, two consumers interested in the same complex events may perceive contradictory outcomes, and thus trigger conflicting reactions. In particular, this makes replicating event consumers difficult for high availability. The manual deployment of proxy consumers, which multiplex complex events to replicated end consumers, and similar pragmatic solutions may work in some special cases. But in general, these tend to shift the problem rather than solve it. In addition, by integrating actions or using compensation into the picture, extending consistency across those actions becomes important. More research is required on models that capture the entire EP application from the production of basic events to the highest-level reactions.

6.4 Privacy and security

In addition to traditional privacy and security concerns, event processing needs to address specific challenges in i) providing access control for consumers and producers of event streams, ii) ensuring the authenticity of events, and iii) providing privacy for consumers as well as producers. The core challenges in event processing systems arise once parts of the event processing infrastructure, such as EPA hosts, cannot be trusted and become apparent with an increasing decentralization of event processing systems.

6.4.1 Access control

Access control mechanisms in event processing need to be provided for producers and consumers, respectively. For consumers, access control ensures that only legitimate data sources and event processing nodes can produce events. Similarly, consumers can only receive those events that have been authorized by the event processing system. Existing work has already proposed solutions for organizing content-based routing and keeping content confidential from unauthorized entities in highly decentralized event processing systems. However, it is a major challenge for access control mechanisms to provide fine-grained access restrictions with respect to the event space of the event processing system and support access control that can dynamically evolve over time. This requires appropriate and scalable key

management to cope with the many possibilities that exist to query events. Furthermore, the key management solutions need to cope with the revocation of keys or limit their time of validity to ensure a dynamic evolution of access control. For highly decentralized event processing solutions, key management yields the main bottleneck. Traditional public/private key infrastructures, if applied in a naïve way, will suffer from a large number of private/public key pairs for the key management system.

6.4.2 Authenticity of events

Once events are mediated over untrusted hosts, it becomes important to ensure the authenticity of these events. Consumers must be able to verify that the source of the event was authorized to publish the event. Again, this is a strong limitation to the naïve use of traditional public/key infrastructure since knowledge is required about the origin of every possible data source. This conflicts with the desired property in many event processing systems of decoupling between producers and consumers. The problem of event authenticity increases if the evaluation and detection of event patterns also cannot be trusted. In this way, additional means are needed to verify both that an EPA operated on the correct input, and that it performed the correct operation required to detect the desired event pattern.

6.4.3 Privacy

Privacy concerns arise for event processing systems to keep the interest of consumers secret. Especially in decentralized event processing systems, all processing entities and consumers contribute to the forwarding of events. To support efficient routing and keep the content of events confidential from unauthorized entities, many systems leak information by exploiting information inherent in the routing tables. Privacy concerns may also occur for publishers in an EP application. If multiple business domains interact, the propagation of a detected event pattern may also leak additional information on data that actually led to the corresponding pattern. Such privacy concerns are likely to conflict with other security concerns like authenticity of events. The research challenge is to define and enact such a privacy model

6.3 Summary

These shorter term research activities can be constructed incrementally over existing models and tools, while the Fabric mentioned in Chapter 2 requires re-thinking of the basic assumptions.

REFERENCES

[Ammon 2009]
Rainer von Ammon, Christoph Emmersberger, Thomas Ertlmaier, Opher Etzion, Thomas Paulus, Florian Springer: Existing and future standards for event-driven business process management. DEBS 2009

[Barga 2007]
Roger Barga et al. Consistent streaming through time: A vision for event stream processing. In CIDR, 2007

[Chandy 2009]
K. Mani Chandy, W. Roy Schulte: Event Processing, Designing IT Systems for Agile Companies. McGraw Hill, 2009

[Dayal 1996]
Umeshwar Dayal, Alejandro P. Buchmann, Sharma Chakravarthy: The HiPAC Project Active Database Systems: Triggers and Rules For Advanced Database Processing, Jennifer Widom, Stefano Ceri (Eds.), Morgan Kaufmann/Elsevier, San Francisco, CA, 177-206, 1996.

[Etzion 2010]
Opher Etzion, Peter Niblett: Event Processing in Action. Manning, 2010

[Eugster 2003]
Eugster, P. T., Felber, P. A., Guerraoui, R., & Kermarrec, A.-M. (2003). The many faces of publish/subscribe. ACM Computing Surveys, 35(2), pp. 114-131.

[Fabret 2001]
Françoise Fabret, Hans-Arno Jacobsen, François Llirbat,Joăo Pereira,Kenneth A. Ross,Dennis Shasha Filtering algorithms and implementation for very fast

[Fidler 2005]
Eli Fidler, Hans-Arno Jacobsen, Guoli Li, and S. Mankovski. The PADRES Distributed Publish/Subscribe System. In Feature Interactions in Telecommunications and Software Systems VIII, pages 12-30, Leicester, UK, July 2005.

[Jacobsen 2009]
Hans-Arno Jacobsen, Vinod Muthusamy, and Guoli Li. The PADRES event processing network: Uniform querying of past and future events. IT - Information Technology, 51(5)250-260, 5 2009

[Jain 2008]
Namit Jain, Shailendra Mishra, Anand Srinivasan, Johannes Gehrke, Jennifer Widom, Hari Balakrishnan, Ugur Çetintemel, Mitch Cherniack, Richard Tibbetts, Stanley B. Zdonik: Towards a streaming SQL standard. PVLDB 1(2): 1379-1390 (2008)

[Jain 2009]
Namit Jain, Shailendra Mishra, Anand Srinivasan, Johannes Gehrke, Jennifer Widom, Hari Balakrishnan, Ugur Çetintemel, Mitch Cherniack, Richard Tibbetts,Stanley B. Zdonik: Towards a streaming SQL standard. PVLDB 1(2): 1379-1390 (2008)

[Li 2008]
Guoli Li, Vinod Muthusamy, and Hans-Arno Jacobsen. Adaptive Content-based Routing in General Overlay Topologies. In ACM Middleware, 2008.

[Li 2010]
Guoli Li, Vinod Muthusamy, and Hans-Arno Jacobsen. A Distributed Service Oriented Architecture for Business Process Execution. ACM Transactions on the Web, 2010.

[LINQ]
The LINQ Project. http://tinyurl.com/42egdn.

[Luckham 2002]
David Luckham: The Power of events. Addison Wesley, 2002

[Muthusamy 2009]
Vinod Muthusamy, Hans-Arno Jacobsen, Tony Chau, Allen Chan, and Phil Coulthard. SLA-Driven Business Process Management in SOA. In CASCON, 2009.

[Muthusamy 2010]
Vinod Muthusamy, Haifeng Liu, and Hans-Arno Jacobsen. Predictive Publish/Subscribe Matching. In ACM Distributed Event-based Systems (DEBS), 2010.

[OMG EDA]
http://soa.omg.org/SOA-docs/EDA-Standards.htm

[RuleML]
Reaction RuleML 1.0 – EP RuleML Language Family - http://reaction.ruleml.org

[Soule2010]
R. Soule et al. A Universal Calculus for Stream Processing Languages. In ESOP, 2010.