A Conceptual Model for Event Processing Systems

Redguides
for Business Leaders

Catherine Moxey
Mike Edwards
Opher Etzion
Mamdouh Ibrahim
Sreekanth Iyer
Hubert Lalanne
Mweene Monze
Marc Peters
Yuri Rabinovich
Guy Sharon
Kristian Stewart

- Learn about a conceptual model to represent event processing systems
- Identify key components of event processing systems
- Discover how the model can be applied to practical examples

Executive overview

This IBM® Redguide™ publication introduces a Conceptual Model for Event Processing. This model is described in terms of an underlying Event Processing Network and an associated Conceptual Architecture for Event Processing, which provides a conceptual view of the event processing architecture and the key components required to build useful event processing systems.

We begin with an introduction to the concepts of business event processing, and its growing importance in the industry, to provide the basis and motivation for a conceptual model.

We provide an explanation of the Event Processing Network, and of our Event Processing Conceptual Architecture and its components, which represent the building blocks that can be used to support an implementation of the Conceptual Model. We describe the flow through the conceptual model, with examples to make this concrete.

Throughout this paper, we illustrate the concepts using practical scenarios. We discuss in some detail three scenarios that are then used to validate the conceptual model, and to demonstrate that the model is sufficient to address the needs of these disparate and non-trivial event processing use cases.

Introduction

Enterprises in many domains have always behaved in an event-driven way and have had to deal with a growing volume of business events and transactions on a daily basis. Event Processing (EP) is an emergent area driven primarily by the greater need of enterprises to respond quickly to this large volume of business and IT events. It recognizes the need to support the decision-making cycle by processing enterprise-significant events more effectively, and is increasingly becoming an important part of enterprise strategies for service-oriented architectures (SOAs).

This paper first discusses the need for event processing, the different types of event processing and the value for businesses of adopting event processing. The paper then details a conceptual model of event processing which can be used to realize an Event Driven Architecture (EDA). The concepts are elaborated by discussing three business scenarios that implement event processing for improved decision making.

Event processing overview

An *event* is anything significant that happens or is contemplated as happening. An event happens completely or not at all and is significant because it may affect some action. It is contemplated as happening because it could be a fact becoming true or could be a transition of an entity in the real world. It might be part of a business process; for example, a trade order has been issued, an aircraft on a specific flight has landed, a reading of sensor data has been taken; or it might be monitoring information about IT infrastructure, middleware, applications, and business processes. Figure 1 provides a high-level overview of event processing.

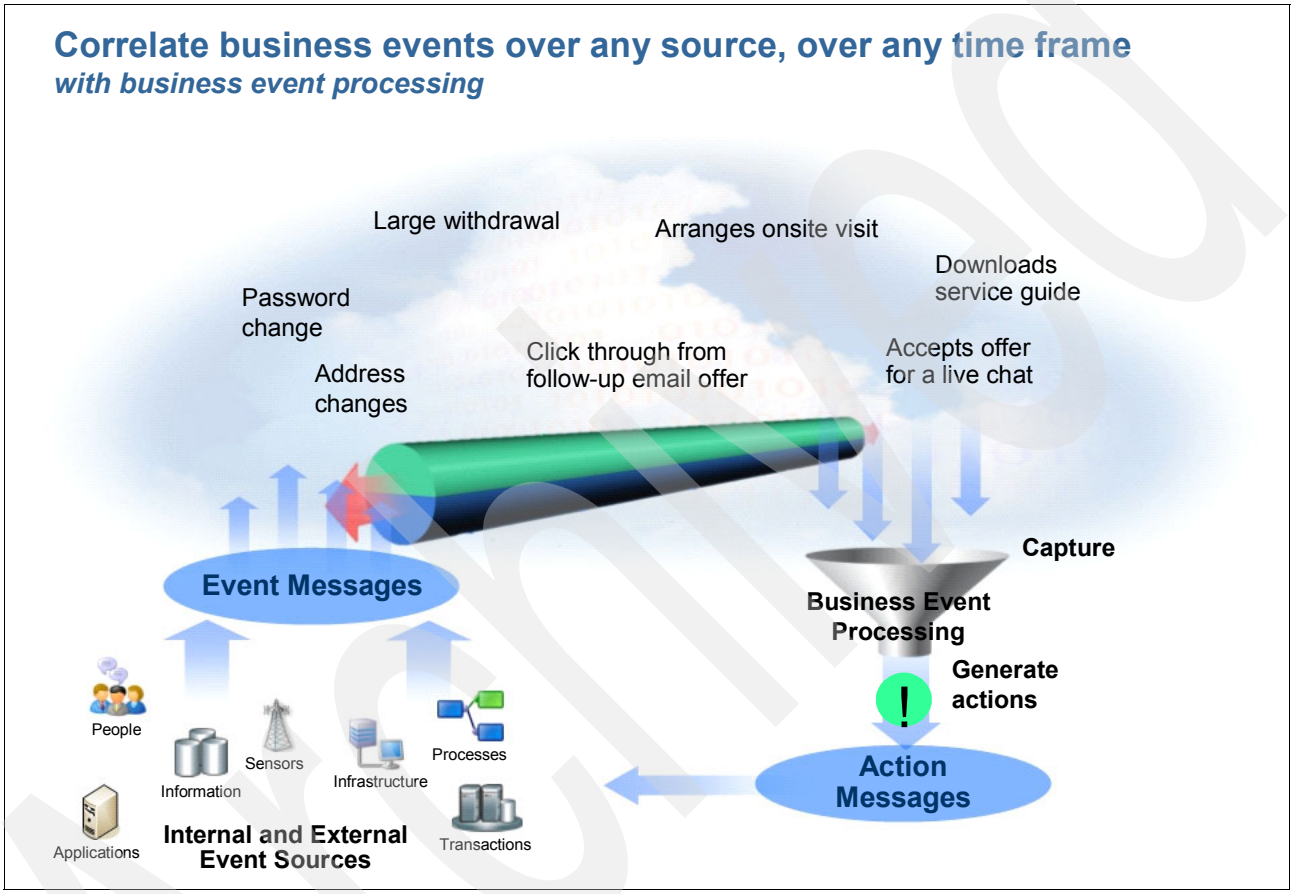

Figure 1 Overview of event processing

An event (a notable thing that happens inside or outside the business) can trigger the invocation of a service, the initiation of a business process, or the publication or syndication of further information. *Event processing* deals with the task of processing one or many events with the goal of identifying the meaningful events within the event cloud. From a business value perspective, event processing is the ability to detect and respond to events that indicate business-impacting situations occurring across the enterprise. For example, an event message could indicate a new customer was added, a product was sold, a shipment was received, a security door was opened, give the current location of an asset via global positioning system (GPS), and so on.

An event producer (or event source) produces events. An event producer might, for example, be an application, data store, service, business process, transmitter, sensor, or a collaboration tool, such as an instant messaging or e-mail application. Upon receipt of the events from the producer, the events can directly lead to outcomes, or be evaluated against event processing patterns. The event processing patterns are defined in accordance with the

needs of the interested parties, not of the event producers. Event processing outcomes include, but are not limited to, invoking a service, initiating a business process, publishing the event out to a subscription hub, directly notifying humans or systems, generating a new event, or capturing the event for historical purposes. Events and their interaction with business services are commonly referred to as an event-driven information system or an event-driven architecture.

Conceptual foundations

The conceptual building blocks of an architecture that supports event processing, that is, an event processing system, should provide core functions such as event-processing logic, and connect event producers and consumers through events. A useful model for thinking about such architectures and systems is the *event-processing network (EPN)* construct, a conceptual formulation that describes the structure of event-processing systems and the common features that they should all support. The EPN describes an event processing system as a collection of interacting event producers, processing agents and consumers. In this context, the primary responsibilities of the EPN are to receive events from producers, pass them on to the correct combination of event processing agents to process the events, and deliver the processed events to the right consumers. The EPN construct is described in more detail in the section “Conceptual model” on page 10. The conceptual model encompasses visualization, event databases, event-driven middleware, event processing languages, and everything that supports the management of events through their lifecycle – from modeling and programming to monitoring and responding.

Types of event processing

Event processing functions can be categorized into simple and complex, broadly determined by processing complexity and sophistication.

- Simple event processing:

  Simple events, meaning events that do not summarize, represent, or denote a set of other events, are filtered and routed without modification. Thus, a notable event happens, initiating one or more downstream actions, and each event occurrence is processed independently. Although referred to as *simple*, such events can provide great value and provide considerable business information. Events are transformed, which entails translating and splitting events, and merging one or more events. Simple processing includes processing such as changing the events schema from one form to another, augmenting the event payload with additional data, redirecting the event from one channel or stream to another, and generating multiple events based on the payload of a single event. This type of event processing is not always distinguished as a separate type.

- Complex event processing:

  Patterns spanning multiple independent events are detected in order to derive new *complex events*. A complex event is an event that summarizes, represents, or denotes a set of other events. Complex event processing includes processing over a set of events in order to detect a business-significant situation. Typically, this processing involves applying a collection of evaluation conditions or constraints over an event set. The events (notable or ordinary) might span different event types and might occur over a specified time period. Events might be correlated over multiple dimensions of interest, including causal, temporal, spatial, and other dimensions. A distinction should be made between the complex and rich nature of the information available from complex event processing, and the fact that it is not, or should not be, complex for the user.

The modeling and implementation of event processing constructs is supported by application integration middleware, as well as a variety of network and system management platforms, although a variety of programming models are used to express these constructs. Complex event processing tools and engines have emerged in recent years. Event processing can be

combined with analytic techniques to predict events, mine patterns, and embed real-time analytics in routing decisions and event derivations. Using these sorts of techniques to do real-time event analysis allows more intelligent decision making within the decision-making loop.

Business event processing

Business event processing extends and builds on event processing, by making it available to the business user in a consumable way (Figure 2). Business event processing extends the capabilities and tools to the business user, allowing the application of the technologies to construct event-driven information systems which contribute to the business. The ability to sense when an event or event pattern has happened (or not happened), which indicates an actionable business situation, enables the business to respond rapidly to opportunities and threats.

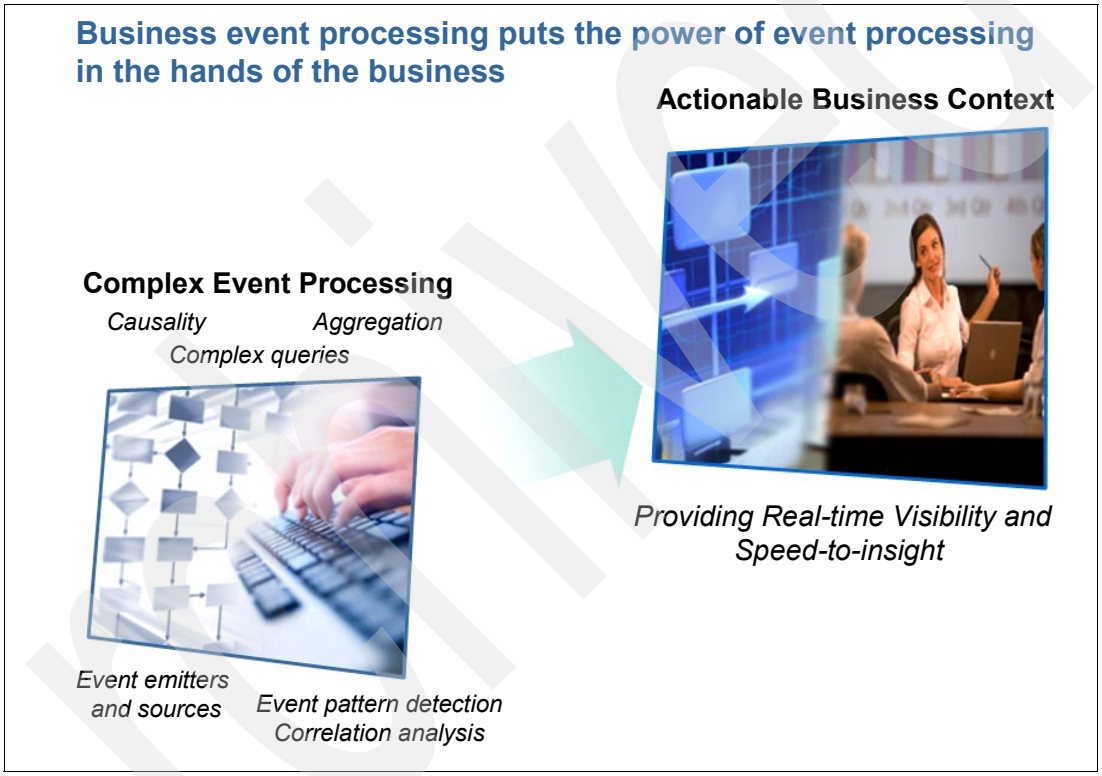

Figure 2 Business event processing adds business perspective to event processing capabilities

Business managers and analysts understand actionable situations, that is, key events or combinations of events and the desired actions to be taken in response to those events. They deal with them every day and are directly responsible for knowing when they occur and managing the response. However, they do not have the solutions available enabling them to identify and respond to the volume and complexity of these situations themselves. At the same time, although millions of potentially actionable events are flowing freely through the IT infrastructure today, supporting advanced event processing functionality requires a specialized set of capabilities, but most IT organizations do not have the technology to effectively and efficiently support these requirements. Business event processing is specifically designed to address this challenge; that is, leverage the information flowing through business systems to support business decision-making processes.

Business event processing is the capability to sense an event, indicate that an actionable business situation has occurred, and coordinate the right response (action) at the right time. Business event processing provides an integrated set of systems and infrastructure that

monitors events happening across the enterprise. This type of processing recognizes significant occurrences as they take place and triggers alerts and disseminates information about the event to initiate appropriate responses. When included as part of a Business Process Management solution, business event processing provides a powerful combination of timely event pattern detection with dynamic business process execution. Business event processing provides IT with the functionality to support advanced event processing requirements in a high-performance, manageable, scalable environment. Additionally, through the inclusion of graphical, nonprocedural user interfaces, business event processing equips business users to define the event processing interactions and actions themselves.

Value of event processing

The basic principles of event processing have been widely used for some time, in application integration middleware and various forms of system software such as operating systems, and network and system management software. However, the increased value of event processing lies in recognizing the significance of an event from a business context, and identifying the right responses to associate with that event. In turn, this approach can allow a business to respond quickly to new opportunities and to competitive threats, disseminate relevant information in a timely fashion to the right people, enable active diagnosis of problems, and contribute to creating a real-time view of the general state of the business.

Event processing can help businesses to identify trends and threats, seize opportunities to mitigate risks, promote faster time to value, and enable rapid sense and respond cycle times. There is a growing market across various industries for event processing, for example:

- Traders in capital markets wanting to react to arbitrage opportunities
- Military or intelligence analysts assessing streams of satellite and sensor data to determine appropriate offensive or defensive actions
- Transportation and logistics businesses utilizing real-time vehicle telemetry to manage vehicle fleets more effectively
- Bankers tracing transactions continuously to look for fraud, money laundering, or breaches of financial regulations
- Providers of communication services seeking to minimize the mean-time-to-repair of faults in the network
- Oil companies determining the depth and breadth of drills dynamically, based upon real-time operational data
- Automobile part suppliers utilizing complex manufacturing decisions to provide parts to manufacturers for *just in time* production

In all these cases and more, there is an inherent requirement to handle large volumes of complex data in real time – and event processing provides this capability. The need for enterprises to move from batch processing to real-time processing for quick decision-making is another factor pushing the demand for event processing. The characteristics of emerging workloads also require close to real-time complex event processing, involving not only data events but also events that originate from non-conventional sources like voice and video.

This view is echoed by industry analysts who take the view that events in several forms, from simple events to complex events, will become much more widely used in business applications. There are enormous financial and strategic benefits to implementing event-driven business processes, because they suit the inherently event-driven nature of many aspects of the real world.

Event-driven business processes are not just traditional processes made to run faster, rather, they have specific characteristics that distinguish them from *business as usual*. Event-driven applications allow processes to be modified rapidly and to respond to errors and exceptional

conditions that disrupt conventional processes. As enterprises strive to cut costs and improve their responsiveness to customers, suppliers, and the world at large, the concept of event-driven design is becoming more widely used. Enterprises are seeing benefits by implementing event-driven business processes, not only because they suit the inherently event-driven nature of business, but also because it gives them a competitive advantage in terms of cost and time to value.

Event processing scenarios overview

To illustrate the concepts expressed in this paper, we describe three different event processing scenarios, which demonstrate some of the value described in the overview.

- Fleet Management scenario
- Public Health Alert scenario
- Communication Service Provider scenario

After introducing the scenarios we map the various event processing concepts to each scenario and explain the added value of event processing for this kind of scenario. The first scenario (Fleet Management) is covered at a greater level of detail because this scenario is where many of the common aspects are introduced.

We begin by describing the business context of the scenarios, and discuss in “Event Processing Network scenarios” on page 13 the events involved and mapping to aspects of the conceptual model.

Fleet Management scenario

This scenario describes a fictitious delivery company that specializes in delivering small packages over relatively short distances, by a fleet of vehicles; each vehicle has GPS that constantly broadcasts its location, and the packages are tagged with radio frequency identification (RFID) tags. Depending on the service level desired, the package is either delivered immediately point to point, or is sent to a hub, where possibly another vehicle will deliver it to its destination. For each package there is a time when it is guaranteed to be delivered, with penalty for being late. Some of the deliveries are fixed (on a monthly schedule), and some arise from customers calling (or using a Web site) to request orders (see Figure 3).

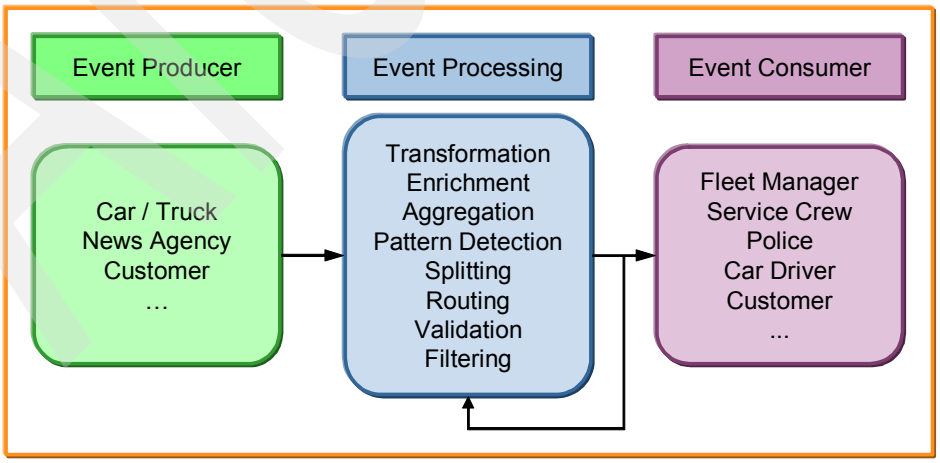

Figure 3 Fleet Management scenario overview

Ideas for this scenario were based on the IBM solutions for fleet optimization[1].

For truck and auto fleet managers, the following represent some of their objectives in managing and maintaining the business:

- Reduce fuel costs
- Track vehicles and drivers
- Provide up-to-the-minute information for customers
- Find the sweet spot to streamline the loading, unloading, and route planning process
- Increase asset utilization
- Improve customer service
- Drive down operating costs
- Reduce unscheduled maintenance costs
- Mitigate risks to reduce insurance costs

In order to manage the fleet in a timely and appropriately way, the company chose to make use of event processing within their system. Thus, the company is able to react quickly, and reassign routes to meet new customer demands as they occur, while keeping its agreement with each customer. The company is also able to react to risks as they materialize and mitigate the situation by making reassignments before agreements are breached. With event processing the company can react to opportunities as well as risks when they materialize and decide how to act (to keep the indicators mentioned under control) because the current situation across vehicles, orders, and so forth, is constantly monitored and brought forward through events. The company is also able to make changes and deploy them into the process quickly and confidently by merely changing the event processing logic or application without going through long and error-prone development processes.

In “Fleet Management scenario details” on page 13 we look at the events involved and the event processing concepts for this scenario.

Public Health Alert scenario

The Public Health Alert scenario describes the alerting system in cases that are indicative of real or potential outbreaks that pose a risk to the population. SARS and avian flu (H5N1), or terrorist attacks using biological or chemical agents, are only a few examples of outbreaks of interest (see Figure 4).

The chosen scenario considers the outbreak of Avian Influenza (Bird Flu) and Avian Influenza A (H5N1), although it could equally apply to other outbreaks such as H1N1.

The stages or states of a pandemic as defined by the World Health Organization show the progression from inter pandemic state (where no influenza viruses are detected in humans) to the pandemic state (where you have sustained transmission in the general population). For the scenario three stages are primarily considered: no detection of virus, epidemic state, and finally pandemic state.

[1] “Fleet optimization for travel and transportation”, IBM, 2009, TTS03009-USEN-00, ftp://ftp.software.ibm.com/common/ssi/pm/sp/n/tts03009usen/TTS03009USEN.PDF

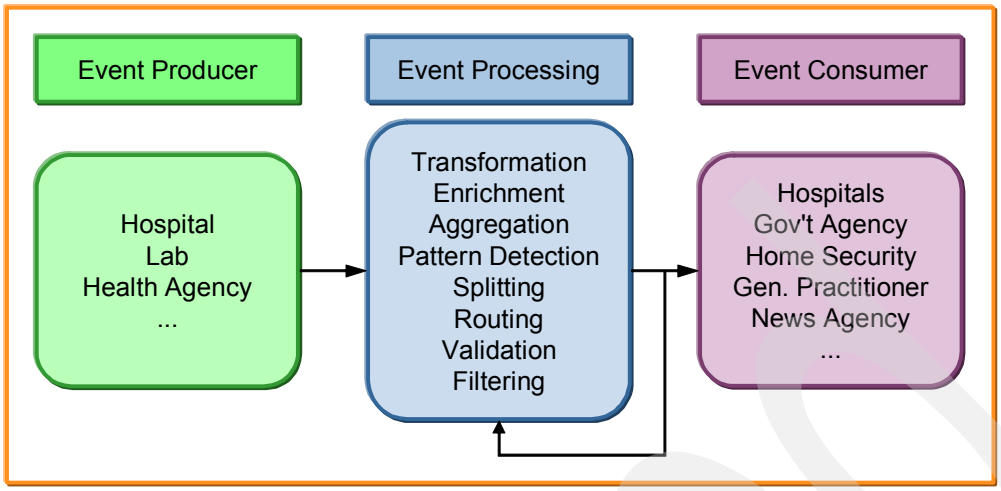

Figure 4 Public health alert overview

The ideas used in this scenario have been extracted from a publicly available IBM Redbooks® publication, *Healthcare Collaborative Network Solution Planning and Implementation*, SG24-6779.

We now describe what happens in this scenario. A laboratory performing blood analysis detects a virus. According to regulations, the detection of a specific virus is published as an event. The event receiver (in our term the event emitter) normalizes the event and performs some basic quality and origin checks before passing it to the event processing agents. where the event is enriched with further information and then pattern detection checks whether an epidemic outbreak alert needs to be submitted. In case of an epidemic outbreak other enrichment and further pattern detection logic can be applied to check for a pandemic outbreak. In the case of epidemic and pandemic outbreak alerts, notifications are sent as corresponding events. Pandemic outbreak alerts from external event producers like international health organizations or foreign governments are treated in a similar way.

The events involved and how this scenario relates to the Event Processing Conceptual Model are covered in the section “Mapping of the scenarios to the conceptual model” on page 32.

Communication Service Provider scenario

This scenario describes “Fictional Communications Service Provider Company A,” a Communication Service Provider (CSP), which is hereafter referred to as “Communications Service Provider Company A.” This company offers a variety of wire line communication services to domestic customers and small to medium sized businesses, ranging from broadband Internet access to Voice over Internet Protocol (VOIP) and Virtual Private Network (VPN) services. The company’s core Multiprotocol Label Switching (MPLS) network is comprised of network elements from many different manufacturers supported by an array of Element Management Systems. They employ a range of network technologies for connectivity with customer premises, including Dial-up, Digital Subscriber Line (DSL), and technologies supporting layer 2 and 3 VPNs.

Depending on the particular service package purchased by a customer, “Communications Service Provider Company A offers varying Service Level Agreements (SLAs). These SLAs are contracts guaranteeing agreed levels of service according to specified metrics, for example continuity of service or available network bandwidth. SLAs that are not honored can lead to financial penalties for the company and can damage its reputation. Customers are tiered according to their individual agreed levels of service.

This company, like other CSPs, employs a Network Operations Center comprised of employees, and hardware and software assets dedicated to the smooth running of the company's core network, and the services it provides. Their objectives include:

- Minimizing mean-time-to-repair of faults in the network that might lead to outages or degradation of services provided to the company's customers.
- Prioritizing the remediation of outages and faults according to the SLAs in place with affected customers, thus minimizing the company's financial exposure.
- Maximizing the utilisation of network assets.
- Minimizing operating costs such as manpower and energy consumption.
- Providing proactive communication with customers in the event of service-affecting faults, and keeping them abreast of progress made to correct said faults.

The company's Network Operations Center uses network management and event processing technologies to these ends. These systems process many types of events, including events that represent:

- Detected faults in the network
- Changes in the state of network elements
- Environmental conditions in the premises that house network equipment and the network equipment itself
- The results of automated responses to events
- The actions of human operators that respond to network events
- Automatic detection of resolution of faults

Access to this event data, and the tools to process it, allow the Network Operations Center to:

- Monitor the elements that comprise the core network for health, availability, and performance
- Normalize network events to a common format for consistent visualization and processing
- Provide up-to-date interactive views of events that have happened in the network to Network Operations Center operators
- Automatically discover and maintain up-to-date models of the core network in order to:
  - Maintain visibility of deployed assets
  - Maintain models of the physical and logical network topology and their relationship with provisioned network services
  - Provide network map dashboards to the operation staff for the purpose of problem diagnosis and troubleshooting
  - Perform network topology based root cause event analysis of network events and service outages
  - Support provisioning process for new services
- Perform automatic response, diagnosis and remediation of a large number of network failure scenarios
- Perform pattern analysis on events and make inferences regarding the causes, and business impact, of network events
- Perform enrichment of network events based on technical, topological and business context
- Define policies for automatic creation of trouble tickets in the company's service desk application, leading to the instigation of physical maintenance activities

Communications Service Provider Company A has a contract with another business “Fictional Virtual Private Network Provider Company X,” referred to hereafter as “Virtual Private Network Provider Company X”. Virtual Private Network Provider Company X provides VPN services between its distributed premises. The contract is subject to high levels of availability with an associated SLA. Communications Service Provider Company A provides and manages a dedicated Customer Edge Router (CE) at each of the VPN provider’s premises for connection to the Communications Service Provider Company A core network. Each CE is connected to a Provider Edge Router (PE) on Communications Service Provider Company A’s premises.

Should a fault occur in the PE router, for example a failed card, the event processing system will receive many events indicating physical and logical faults sent from the equipment and element management systems in the surrounding network, including Internet Control Message Protocol (ICMP) ping failures, link down alarms, and routing protocol adjacency failures. In this situation, the event processing system must:

- Identify the root cause event, in this case the card failure, and highlight it to a Network Operations Center operator.
- Infer whether any customer services are affected, in this case, whether Virtual Private Network Provider Company X’s VPN service is rendered inoperable.
- Determine the relative priority of the service outage according to applicable SLAs versus other outages in the network and appropriately prioritize problem resolution. In this case, the resolution is high priority due to the aggressive SLA with Virtual Private Network Provider Company X.
- Inform the affected customer that the fault has been identified, and raise a work item request to dispatch a technician to address the fault.
- Detect the resolution of the fault, and inform the operator and customer that normal service has been re-established.

The events involved in this scenario and mapping to the conceptual model are considered in in the section “Communication Service Provider scenario details” on page 19.

Conceptual model

Our conceptual model presents two different views of event processing systems, aimed at describing the important concepts and their relationships at an abstract level, removed from any technical details. The Event Processing Network (EPN) abstracts the essential features of the input, processing, and output elements of an event processing system. Guided by EPN concepts, our conceptual architecture identifies the abstract architectural elements, or components, that can be involved in realizing an event processing system, and the inter-relations between them, in order to deliver business value. This model is at an abstract level that is independent of the technologies, protocols, and products that could be used to provide the components.

The goal of the conceptual architecture is to form the basis for implementations of event processing systems and event-driven applications, and to provide a common framework for specifying, comparing, and contrasting event processing solution architectures and implementations.

It is our intention that the conceptual architecture provides a sufficient set of components at a conceptual level, from which an implementation of an event processing system can be constructed, but there is no requirement to implement any components that are not needed in a given system, nor is there any implied concept of compliance to the model.

Event Processing Network

Our high-level abstraction of event processing systems applies concepts from the event processing network construct (see Figure 5). As stated earlier, an Event Processing Network is a conceptual formulation that describes the structure of event-processing systems and the common features that they should all support [2].

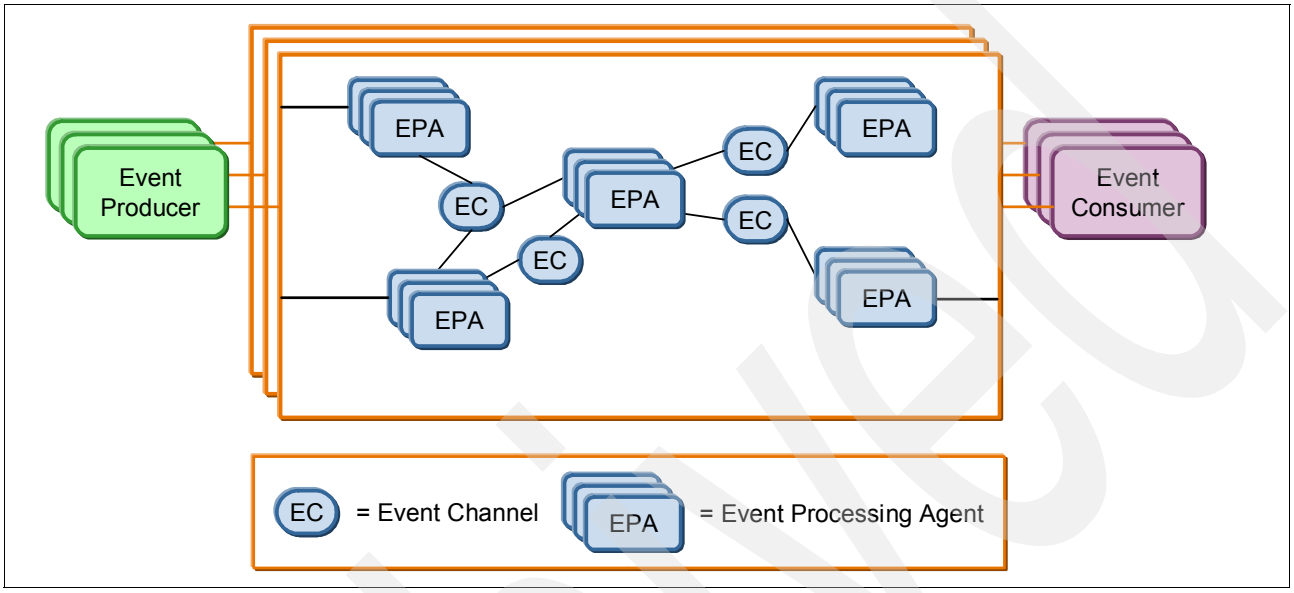

Figure 5 Event Processing Network and its components

An EPN consists of four components:

- Event producer
- Event consumer
- Event processing agent (abbreviated here as EPA)
- A connection component called an *event channel*

An EPN describes how events received from producers are directed to the consumers through agents that process these events by, for example, performing transformation, validation, or enrichment. Any event flowing from one component to another must flow through an event channel, as depicted in Figure 5, which shows the relationship between event channels, consumers, producers, and processing agents. Event channels are nodes which connect event producers to EPAs within the EPN, connect EPAs together as needed, and connect EPAs to consumers, resulting in events passing from event producers through the EPN to event consumers. Figure 5 also illustrates that the various events produced by event producers in the EPN can be processed by appropriate groups of EPAs, connected through channels, in order for those events (or events derived from them) to be consumed by the various event consumers in the EPN.

Event channel

An *event channel* is a mechanism for delivering events or event streams from event producers and EPAs to event consumers and EPAs. At this level of abstraction, no constraints are placed on the properties of the event channel (such as whether each channel can move more than one event type) nor on the mechanism that moves the events.

[2] Etzion, O. and G. Sharon, “Event Processing Network and Implementation”, IBM Systems Journal, Vol 7 No 42 (2008)

The event channel might receive multiple events from different event producers and EPAs, and can transfer events combined from a number of sources to multiple EPAs and consumers. The ordering among the events from different EPAs and event producers to create the combined set of events is implementation-specific and is not covered by the conceptual model, and in some cases no particular ordering is required. However, it is the responsibility of the event channel to take events from the event producers or from the EPAs, ordered (if required) and combined, and provide this to the appropriate event consumers or EPAs.

Another responsibility of an event channel is to retain the history of the events flowing through for retrospective event processing, according to retention policies that determine the duration and filtering conditions for any retained events. Retrospective event processing is the discovery of event patterns over the history of events, as opposed to online event processing, which detects predefined event patterns as new events become available.

An event channel is represented in the EPN as a node with edges directed to and from the node. Every incoming edge represents events from an event producer or event processing agent that places events on the channel; every outgoing edge represents events to an event consumer or processing agent that receives events from the channel.

Event producer and consumer

An *event producer* produces events through event channels for any party of interest to consume. The party of interest can be an event consumer or an EPA. The conceptual model does not place any restriction on the mechanism by which events are obtained from an event producer or EPA, which can involve *push* or *pull* models.

Within a network of nodes and edges, an event producer is represented as a source node, that is, there exist only edges directed from it. The number of edges directed from it is the number of different event channels involved in moving events from the producer to the EPAs or consumers that will receive the events. The same event can be published or made available by the event producer to more than one event channel; however, as a design practice, it is better to leave the routing decisions to the EPAs, for better control, design, and understanding of the overall event processing needs.

An event consumer is interested in events that allow it to perform its responsibilities. Once the event of interest is received by the event consumer, the consumer will perform a certain task or tasks associated with this event.

An event consumer is represented as a sink point, that is, only edges are directed to it. The number of edges directed to it is the number of different event channels involved in moving events to this consumer. The same event can be received through more than one event channel; however, the logic of where, when, and how to receive events is best left to an EPA for better control, design, and understanding of the overall event processing needs.

This is not to say that an event consumer cannot also be an event producer, but when performing the latter role it would appear as an event producer within the EPN.

Event processing agent

In a distributed and heterogeneous system, event producers might not produce the events an event consumer expects to receive. These events might differ in the expected syntax (structure), semantic meaning, or both. There are also cases in which a single event will not trigger an action performed by an event consumer, but instead the action is triggered by a complex composition of events happening at different times and in different contexts. EPAs, also sometimes know as event mediators, are needed to detect patterns in raw events, to process these events through enrichment, transformation, and validation, and to derive new

events. An EPA is responsible for producing these derived events and decides where and how these events should be made available.

The EPA has three possible stages:

- Pattern matching:

  If required, this stage is responsible for selecting events to be processed according to a specified pattern. An EPA that carries out pattern matching is known as a *pattern detection EPA*.

- Processing:

  If required, this stage is responsible for applying the processing functions to the selected events that satisfy the pattern, resulting in derived events.

- Emission:

  This stage is responsible for emitting the events or derived events to a channel.

An EPA receives events from event channels for its pattern detection or other processing, much like an event consumer receives events from event channels for acting upon events. An EPA sends events via event channels when emitting events, much like an event producer sends the events that it produces via an event channel. Within a network of nodes and edges, an EPA is represented as a node with edges directed to and from the node. The number of edges directed to it is the number of different event channels that the agent uses for its functions, such as detecting patterns. The number of edges directed from it is the number of different event channels over which the agent sends events based on the processing and emission definitions.

In summary, an Event Processing Network (EPN) is a directed graph of event processing operations connected through event channels. The EPAs within the network provide event processing services and mediations; that is, get a set of one or more events as an input, perform some processing, and return a (possibly new) set of zero or more events as output. The primary responsibility of an Event Processing Network is to receive events from producers, pass them on to the combination of event processing agents to process the events, and deliver the events to the right consumers.

Event Processing Network scenarios

Let us now map the event processing network definition and components to the scenarios described previously.

Fleet Management scenario details

The following are the event processing network components (producers, consumers, event channels, event processing agents, and also the events) that describe just one of the aspects in tracking the vehicles and drivers introduced in the section “Fleet Management scenario” on page 6. The company has some regulations in place to ensure safety of their drivers and to avoid accidents; therefore, a driver’s shift must not be longer than 8 hours. Any driver exceeding this time is forced to rest and, to avoid delays in delivery to customers and paying penalties, the deliverables must be reallocated to other drivers. The reallocation process requires an appropriate meeting place to be established in one of the company’s facilities, to transfer from one vehicle to another (or others), and to recalculate routes such that the delay is minor, if any. A driver’s report system exists from which start and end events of drivers’ shifts are produced. Based on these events and constant awareness of vehicle locations, the reallocation process can be performed. In addition, there are some requirements for event enrichment, detection of the *exceeding driving time* pattern, and routing, described as event processing agents.

Table 1 identifies the event producers related to the Fleet Management scenario.

Table 1 Event producers for the Fleet Management scenario

Event producers	Description
Vehicle	GPS broadcasting position, on-board sensors (fuel, emission, weight, and so forth)
Driver Report System	Electronic punch card
Delivery System	Order management, package sorting and allocation, and vehicle dispatching system
Package RFID System	Package tracking system with readers at hubs and mobile readers for personnel (for example, used by driver when collecting, transferring, and delivering packages)

Table 2 defines the event consumers related to the Fleet Management scenario.

Table 2 Event consumers for the Fleet Management scenario

Event consumer	Description
Driver Display	On-board and mobile driver display of routes, delivery changes and alerts
Delivery Management Dashboard	Complete view of the operations - vehicle location, routes, orders, packages, and so forth
Clients	Order senders or receivers can receive notifications or look up their orders

Table 3 defines the event types related to the Fleet Management scenario. Figure 6 on page 17 shows the flow of these events by event type.

Table 3 Event types for the Fleet Management scenario

Event type ID	Event type	Attributes	Comments
E1	Driver starts working	Timestamp; Driver ID	
E2	Driver starts working enriched	Timestamp; Driver ID; Vehicle ID; Driver Name	
E3	Driver ends working	Timestamp; Driver ID	Can be based on structure of E1
E4	Driver exceeded driving time	Timestamp; Driver ID; Vehicle ID; Driver Name	
E5	Delivery change	Timestamp; Driver1 ID; Vehicle1 ID; Driver2 ID; Vehicle2 ID; Sender ID; Receiver ID	

Table 4 defines the event channels related to the Fleet Management scenario.

Table 4 Event channels for the Fleet Management scenario

Event channel ID	Event type	Producers	Consumers	Retention policy
EC1	E1	Driver Report System	A1	
EC2	E2	A1	A2	
EC3	E3	Driver Report System	A2	
EC4	E4	A2	A3, Delivery Management Dashboard	
EC5	E5	A3	A4	
EC6	E6	A4	Clients, Driver Display, Delivery Management Dashboard	1 week

Table 5, Table 6 on page 15, Table 7 on page 16, and Table 8 on page 16 define the Event Processing Agents related to the Fleet Management scenario. Figure 6 on page 16 shows the agents in the EPN for the Fleet Management scenario.

Table 5 defines EPA A1 for the Fleet Management scenario.

Table 5 Event Processing Agent A1

Agent Property	Specification
Agent ID	A1
Agent name	Enrich driver starts working
Agent type	Enrich
Agent context	not applicable
Input events	E1
Agent specification	Enrich by driver ID with vehicle ID and driver name from driver details
Output events	E2
Agent comment	Enriches the event with driver details such as vehicle ID

Table 6 defines EPA A2 for the Fleet Management scenario.

Table 6 Event Processing Agent A2

Agent property	Specification
Agent ID	A2
Agent name	Driver exceeded driving time
Agent type	Detect pattern
Agent context	Interval (E2, +8h) by driver ID
Input events	E3
Agent specification	Absence of E3
Output events	not applicable

Table 7 defines EPA A3 for the Fleet Management scenario.

Table 7 Event Processing Agent A3

Agent property	Specification
Agent ID	A3
Agent name	Delivery reallocation
Agent type	Detect pattern
Agent context	not applicable
Input events	E4, Ex
Agent specification	Enrich by Driver ID with Vehicle ID and driver name from Driver Details
Output events	E5
Agent comment	This agent receives different types of events to infer delivery change requirements. It is the responsibility of this agent to decide which vehicle should take over the delivery and how to get the two vehicles to meet at a hub or another location.

Table 8 defines EPA A4 for the Fleet Management scenario.

Table 8 Event Processing Agent A4

Agent property	Specification
Agent ID	A4
Agent name	Route reallocation notification to required consumers
Agent type	Router
Agent context	not applicable
Input events	E5
Agent specification	not applicable
Output events	E5
Agent comment	Routes the delivery change to the consumers. Clients can be notified (will be retried for 1 week), drivers will get their new instructions, and the operations will be able to track these new instructions.

Figure 6 shows the components of this EPN. The Driver Report System produces event E1 and publishes it on channel EC1 as a driver starts his work. Event E1 is enriched by the Event Processing Agent A1 and the derived event E2 is produced. After 8 hours agent A2 produces event E4 because the driver exceeded his driving time before he ended his shift. Event E4 is received by the Delivery Management Dashboard for control. Agent A3 receives this event as well through channel EC4 and reallocates the driver's orders to other drivers to deliver in time but yet avoid exceeding the driving times of other drivers. The reallocation instruction is notified through event E5, which A4 routes and redirects to several consumers that require notifications - the driver who exceeded his driving time, the driver or drivers that need to pick up and deliver his orders, the client to know of changes in the delivery route and expected time of arrival, and finally the Delivery Management Dashboard for control. Only when the driver transfers his deliverables to the other drivers, does he end his shift and event E3 is produced (this event has no effect on the described agents).

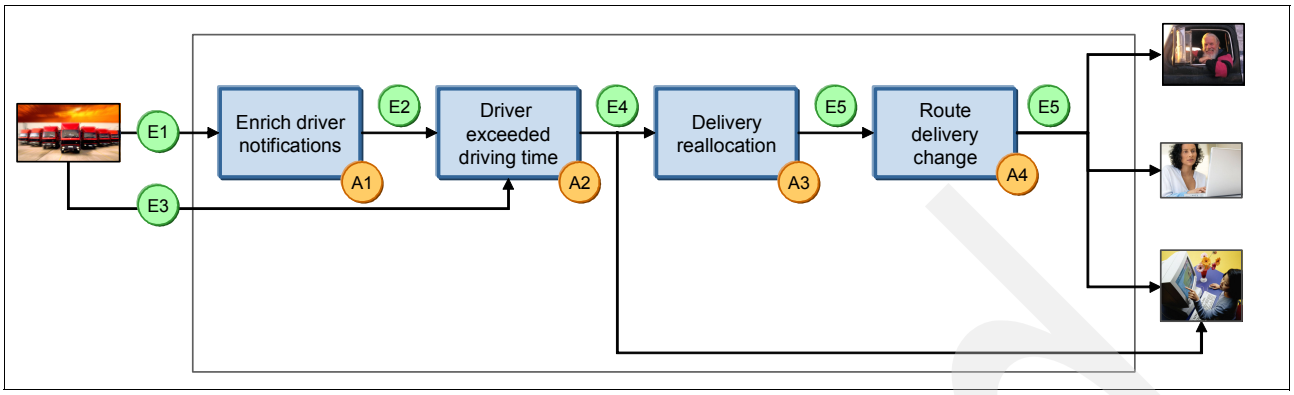

Figure 6 The EPN for the Fleet Management scenario

Public Health Alert scenario details

The following are the event processing network components of the Public Health Alert scenario that was described in the section “Public Health Alert scenario” on page 7. The EPAs for this scenario are not described in detail.

Table 9 identifies the event producers related to the Public Health Alert scenario.

Table 9 Event producers for the Public Health Alert scenario

Event producer	Description
Hospital	The hospital may detect possible viruses or signs and emit an event
Laboratory	The laboratory may detect possible viruses or signs and emit an event
Physician	The physician may detect possible signs and emit an event
Foreign government agency	The foreign government agency emits an event

Table 10 identifies the event consumers related to the Public Health Alert scenario.

Table 10 Event consumers for the Public Health Alert scenario

Event consumer	Description
Pharmaceutical company	The pharmaceutical company subscribes to health alert events.
Government agency	The government agency subscribes to health alert events.
General practitioner	The general practitioner subscribes to health alert events that possibly get enriched with more details on characteristics and detection patterns.
Health insurer	The health insurer subscribes to health alert events.
News agency	The news agency subscribes to health alert events.
Homeland security	The homeland security subscribes to health alert events that possibly get enriched with specific precaution information and detection patterns.

There is also the notion of a kind of man-in-the-middle that implements the event processing network and provides the decoupling between the event producers (who do not know who will consume the event, nor what will be done with the event) and the event consumers (who ignore the origin of the event and the original form or meaning of the initial event).

Beside this relationship between the event producer and the EPN, and the consumer and the EPN, we can also imagine cases where the producers make the event publicly available (via a Web page or feed) and where the EPN retrieves this information. On the other hand, the EPN also makes the event publicly available (maybe via the same mechanisms) for the event consumer. With that in place, there is a decoupling between the producer and the EPN and the consumer and the EPN, and a direct decoupled scenario without the man-in-the-middle can also be envisaged.

Table 11 identifies the event types related to the Public Health Alert scenario.

Table 11 Event types for the Public Health Alert scenario

Event type ID	Event type	Attributes	Comments
E1	Potential epidemic outbreak alert	ID; Details	Converse event is no potential epidemic outbreak
E2	Potential epidemic outbreak normalized	ID; Timestamp; Location; Details	Normalized/transformed event from the original event
E3	Potential epidemic outbreak enriched	ID; Timestamp; Location; More Details	Enriched event
E4	Epidemic outbreak detected alert	ID; Timestamp; Location; Details	Epidemic outbreak has been detected (pattern detection)
E5	Epidemic outbreak alert	ID; Timestamp; Location; Details	
E6	Potential pandemic outbreak alert	ID; Timestamp; Location; Details	
E7	Potential pandemic outbreak normalized	ID; Timestamp; Location; Details	Epidemic outbreak has been detected (pattern detection)
E8	Potential pandemic outbreak enriched	ID; Timestamp; Location; More Details	Enriched event
E9	Pandemic outbreak detected alert	ID; Timestamp; Location; Details	Pandemic outbreak has been detected (pattern detection)
E10	Pandemic outbreak alert	ID; Timestamp; Location; Details	

Figure 7 shows the EPN for the Public Health Alert scenario, showing the events and EPAs involved.

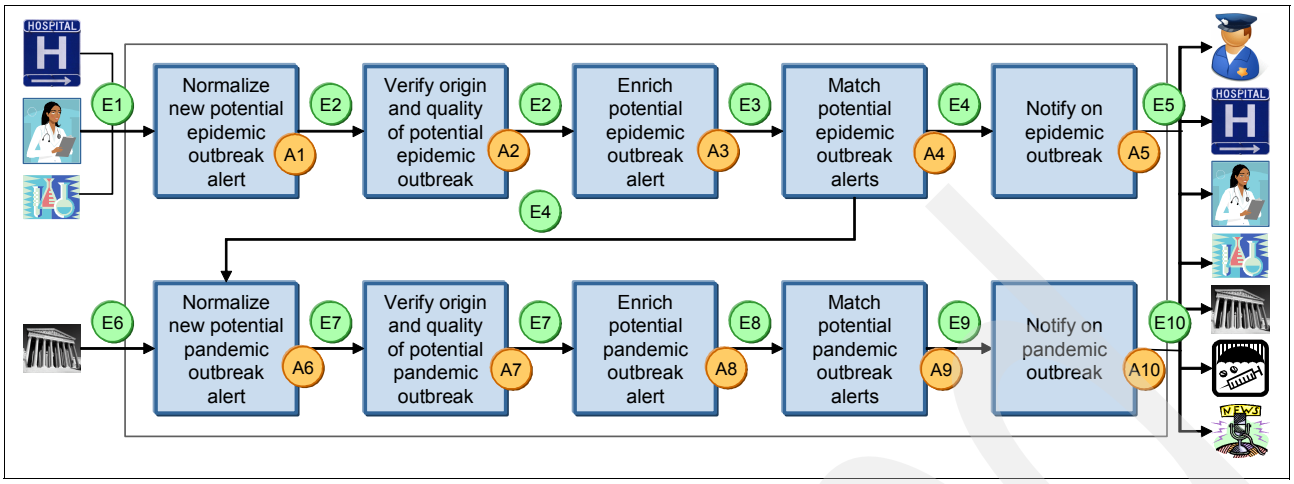

Figure 7 EPN for the Public Health Alert scenario

Communication Service Provider scenario details

The following describes a subset of the event processing network components (event producers, event processing agents, event consumers, and events) in the Communication Service Provider scenario that was described in “Communication Service Provider scenario” on page 8.

Table 12 identifies the event producers related to the Communication Service Provider scenario.

Table 12 Event producers for the Communication Service Provider scenario

Event producer	Description
Network monitor	Software that polls network equipment for availability and performance data, typically using protocols such as ICMP and SNMP. Generates events that represent notable changes to network elements, and exceptions to normal operation.
Network element probes	Receive alerts from network elements and generate events that represent notable changes and faults in the network element and its physical and logical neighbors. Typically uses protocols such as SNMP to listen for alerts.
Element management system probe	Receives alerts from Element Management Systems representing faults and notable changes in the network elements under management. May use standard or proprietary event formats over a wide variety of standard or proprietary protocols including SNMP, SOAP, CORBA, flat file.
Event operator dashboard	Produces events based on operator actions, including the acknowledgement, resolution, and clearing of events from network equipment and service outages.

Table 13 identifies the event consumers related to the Communication Service Provider scenario.

Table 13 Event consumers for the Communication Service Provider scenario

Event consumer	Description
Operator dashboards	Interactive view of significant events to Network Operations Center operators, including customizable diagnostic tools and filtering capabilities.
Network engineer telephone handsets and e-mail system	Recipients of SMS messages and e-mails resulting from critical events that have been escalated by the system.
Customer views	Up-to-date read-only views of detected outages in services filtered for affected customers.
Trouble ticket system	Receives events from the system that require action from the network engineering and maintenance team.

Table 14 identifies the event types related to the Communication Service Provider scenario. Figure 8 on page 22 shows the event types in the EPN for this scenario.

Table 14 Event types for the Communication Service Provider scenario

Event type ID	Event type	Attributes	Comments
E1	Ping failed	Timestamp; Unreachable address	
E2	Link down	Timestamp; Port name of down link on PE router; Address of PE router	
E3	Card failure	Timestamp; Address of PE router; Card identifier	
E4	Root cause event	As E3	Copy of E3, identified as root cause
E5	Symptom events	As E1 and E2; Associated root cause	Copies of E1 and E2, identified as symptoms
E6	VPN service affected	Timestamp; VPN Identifier	
E7	VPN service affected enriched	As E6; Customer Contact Details	
E8	Technician dispatched	Timestamp; Technician contact details	
E9	Ping success	Timestamp; Pinged Address	
E10	Link up	Timestamp; Port name of re-established link on PE router; Address of PE router	
E11	Card up	Timestamp; Address of PE router; Card identifier	
E12	VPN service active	Timestamp; VPN Identifier	

Table 15, Table 16 on page 21, and Table 17 on page 22 identify three of the EPAs for the Communication Service Provider scenario. Figure 8 on page 22 shows the agents in the EPN for the Communication Service Provider scenario.

Table 15 identifies EPA A2 for the Communication Service Provider scenario.

Table 15 Event Processing Agent A2 for the Communication Service Provider scenario

Agent property	Specification
Agent ID	A2
Agent name	Service impact analyzer
Agent type	Pattern detection
Agent context	
Input events	E4, E5
Agent specification	Generate Service Affected event if customer's network service is disrupted
Output events	E6
Agent comments	Uses modelled relationships between network elements and provisioned services to determine whether service is affected

Table 16 identifies EPA A3 for the Communication Service Provider scenario.

Table 16 Event Processing Agent A3 for the Communication Service Provider scenario

Agent property	Specification
Agent ID	A3
Agent name	Customer impact analyzer
Agent type	Routing, enrichment
Agent context	
Input events	E6
Agent specification	Escalate Service affected event according to policy associated with SLA; Enrich with customer contact details
Output events	E7
Agent comments	

Table 17 identifies EPA A4 for the Communication Service Provider scenario.

Table 17 Event Processing Agent A4 for the Communication Service Provider scenario

Agent property	Specification
Agent ID	A4
Agent name	Prioritization module
Agent type	Routing, enrich
Agent context	
Input events	E7
Agent specification	Enrich events with relative priority
Route events to consumers dependent on priority	
Instigate corrective action via trouble ticket system	
Output events	E4, E5, E7, E8
Agent comments	

Figure 8 shows part of the event processing network used in the scenario, which shows up to the point that the technician is dispatched.

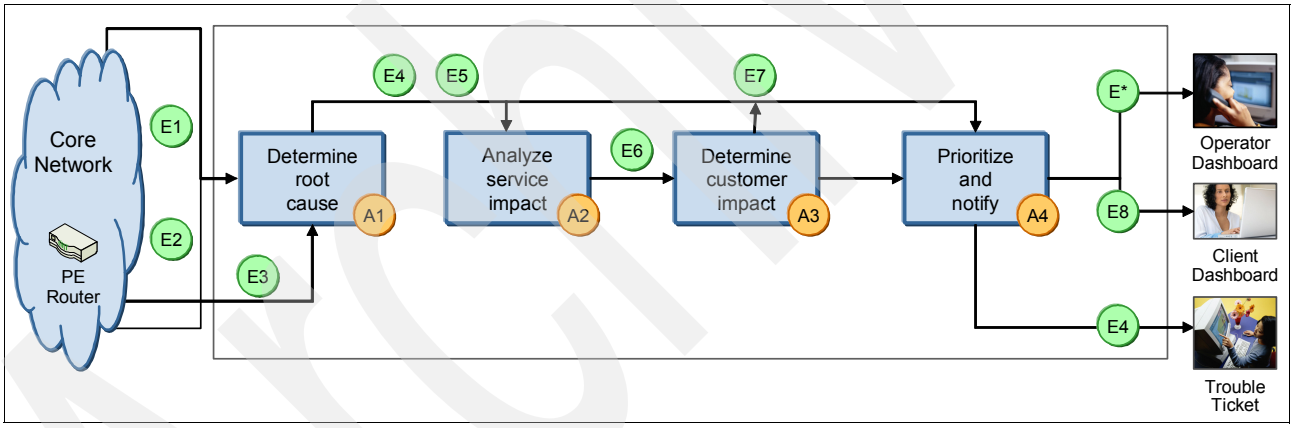

Figure 8 EPN for the Communication Service Provider scenario

Conceptual architecture

The conceptual architecture builds on the concepts of the EPN by defining the components that can be involved in an event processing solution. Some of these components are equivalent to an EPA, or a set of connected EPAs, at the EPN level, while others are more involved in the flow of events and are equivalent to channels in the EPN. Later in this section we show how the conceptual architecture maps to the EPN (see Figure 11 on page 28).

Any system architecture that supports business event processing should enable flexible definition of the event-processing logic: detection of event patterns, derivation of new events, transformation, and routing from producers to consumers based on the business logic required. Thus businesses can react to changes, execute the relevant processes, and influence ongoing processes based on the changes. Furthermore, such a definition of event processing should be easy to modify and quick to deploy in accordance with business needs, such as changes in business processes and policies.

In order to understand how this business value can be derived from event processing systems, it is important to consider a further layer of granularity than the event processing network, to consider components that can be used to construct an event processing system, and their interactions. The outcome of this process is what we term a conceptual architecture for event processing systems. In addition to the three typical layers of an event-driven system (the event producer and associated components, event consumer and associated components, and an intermediary Event Bus layer), the conceptual architecture needs to include components for security, monitoring, analytics, and management of events and event flows.

At the simplest level, a minimal set of conceptual components required for an event processing system consists of an Event Emitter layer to emit events from event producers, an Event Bus, and an Event Handler layer to handle events for consumption by event consumers. This architecture is illustrated in Figure 9, which also includes some examples of event producers and event consumers.

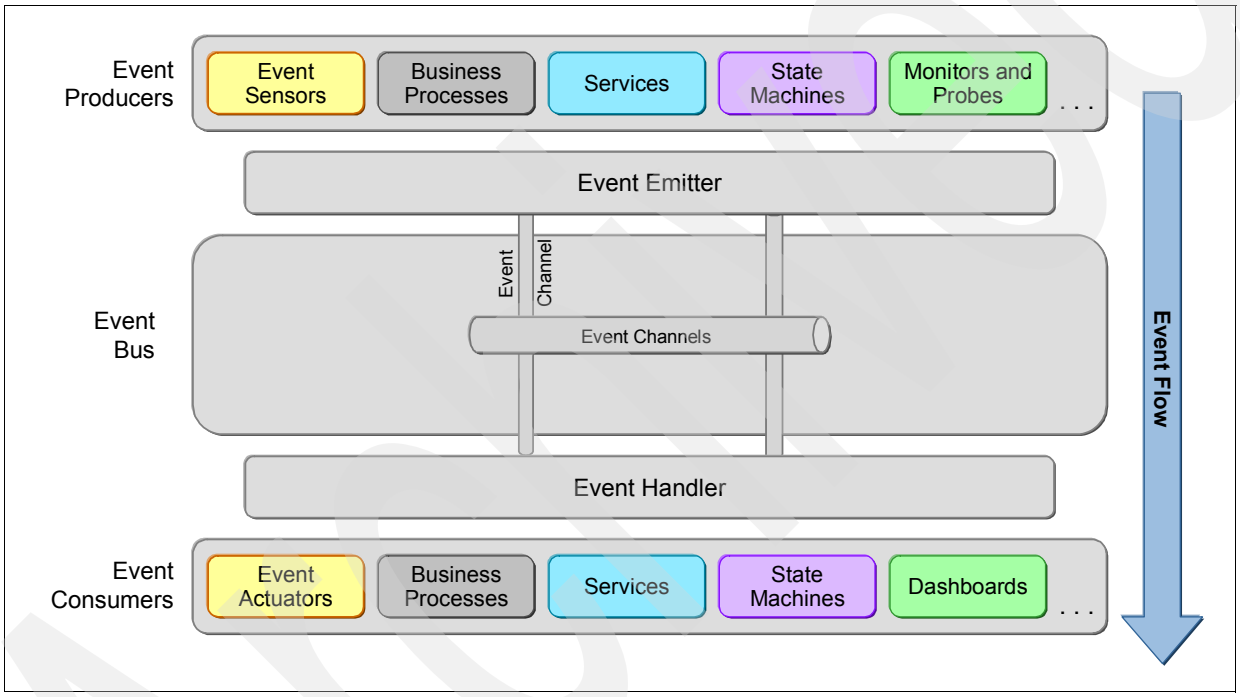

Figure 9 Minimal Event Processing Conceptual Architecture

There might be a need for additional capabilities within the Event Emitter layer, the Event Bus, and the Event Handler (or receiver) layer. In practice, generated events from an event producer cannot always be immediately shared with event consumers. Because the event producer should not require awareness of the event consumers in an event processing system, there is typically a need for a middleware layer between the producer and consumer. This middleware layer performs additional event-related tasks, as well as enabling the consumer to receive those events, or derivatives of them, that are of interest. Not all events are generated by the producer in the required format, and in such cases the events need to be transformed to the required (enterprise standard) format prior to being published to the intermediate layer. In some cases, an ordinary event may be evaluated for notability by an event pre-processor (router, filter), resulting in the generation of a new notable event. Also an event producer may choose not to emit all of the events. Event processing agents can filter and mediate raw events within the domain of the producer.

Similarly, not all the events received at the consumer side may be in a ready-to-use form, so there might be some processing and mediation required at the consumer end. An ordinary

event might be evaluated for notability by an event pre-processor (filter) before detailed handling at the consumer end. The consumer might choose to ignore some of the events it receives. Event processing services fulfill these pre-publish and pre-receive event processing requirements at the event handler layer.

Figure 10 shows all of the components of the Event Processing Conceptual Architecture. Any event processing implementation should be achievable with this as the base set of components at the conceptual level, but that is not to say that every component will be required in every implementation. Similarly, not all of the components will be required for any given scenario. When we return to the scenarios and see how they map to the conceptual architecture, it will more commonly be the case that not every component is involved.

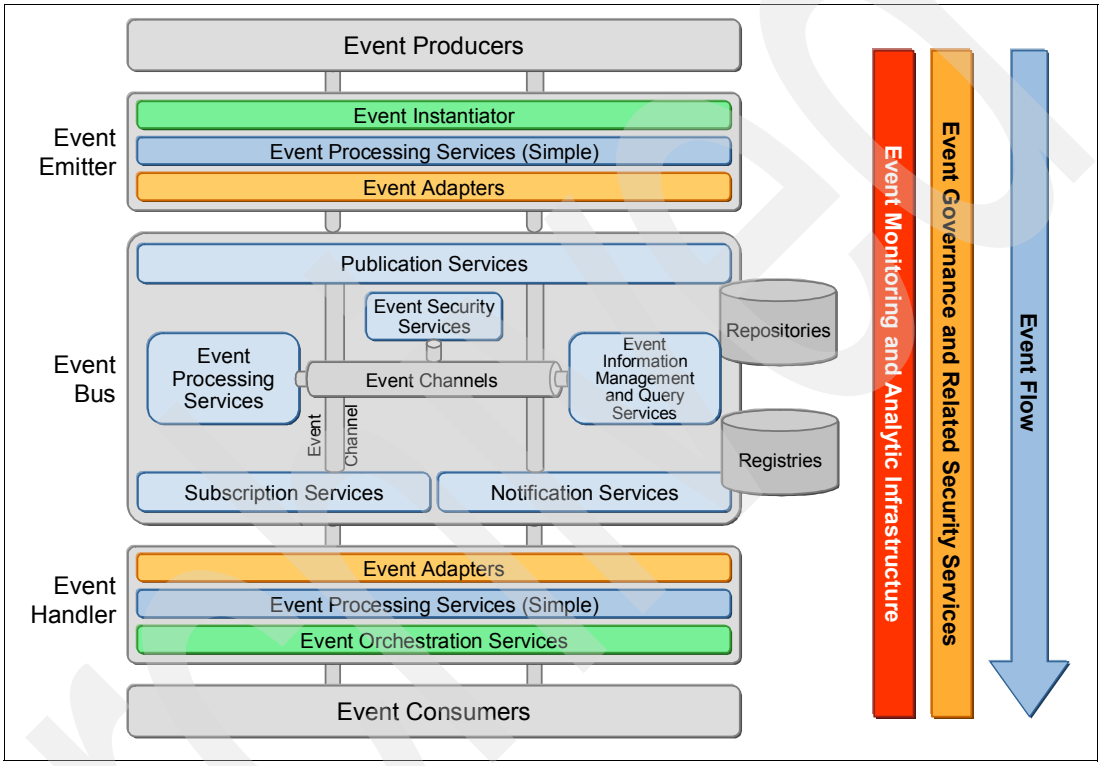

Figure 10 Event Processing Conceptual Architecture: Components which can be involved in an Event Processing system

Architectural components

The event flow in the conceptual architecture is from producer to consumer, and the components shown in the conceptual architecture diagram are summarized here in that order. The section “Flow of processing in the conceptual architecture” on page 28 provides a more detailed description of the flow of processing in the conceptual model. At the implementation level, event consumers will often also be producers of events, but at a conceptual level the roles of consumer and producer are separate.

- Event producer

  The event producer emits an event when something of interest happens (or does not happen). The event producer does not include logic for manipulating events, nor any decision logic on what to emit when, and the events that are generated could be redundant or irrelevant. Typical examples of event producers include:

  – Event sensors, which detect situations (things that happen) and generate raw events, or originate events from data streams or business flows - transmission of temperature is one example

- Monitors and probes, which produce events about availability and problems in systems, such as faults in an IT network
- Business processes, which produce events at significant points in the processing (for example, at milestones or checkpoints), or when a specific process task is reached or started
- Services and applications, which produce events at key points in the processing, such as when the service is invoked and completes, or when it fails
- State machines, which generate events when changing state

- Event emitter

  The event emitter is logically (although not necessarily physically) associated with the event producer, and is responsible for converting and packaging raw events from producers, for delivery to the event bus. The event emitter can include an event instantiator, which creates the instances of events; simple event processing services, such as filtering and mediation of events emitted by a single producer, enriching the event with information available at the time the event occurs; and event adapters. The event instantiator takes events from the producer and does whatever (if anything) is needed to make it available for further event processing or delivery, which can include aggregation, caching, and serialization of events. The event instantiator might be required to manipulate the event header to embed *semantic metadata* into the event message itself, and to make it self-describing (with information such as the time, date, instrument type, process ID, and so forth). The event emitter can provide optimization by carrying out simple event processing at this stage rather than after events reach the event bus. Event adapters can provide formatting and protocol conversion of the event into a form to be received by the event processing network, such as wrapping event records as event messages and sending the event messages to the event bus.

- Event bus

  The event bus receives events from event emitters, potentially at a very high volume of events, and invokes consumers via event handlers as a result of events. Among the capabilities of an event bus can be processing to derive a lower volume of more informative events from the incoming events. The components of the event bus do not have to be co-located. The section “Event bus components” on page 26 gives further detail about the event bus.

- Event handler

  This component prepares the events from the event bus for consumption by the event consumer, receiving events and deciding how to react. The event handler has event adapters to receive event messages from the event bus and unwrap them to get event records. The event handler can also provide simple event processing services, which carry out consumer-side processing to filter and mediate events received from the event bus. Event handlers can also determine the appropriate consumers to react to an event, and invoke the consumers with context derived from the event. Finally, an event handler can provide event orchestration services to manage the distribution of events to consumers.

- Event consumer

  The event consumer performs tasks in reaction to an event. The event consumer has limited concern about the origins of the event, and is just aware that it is being invoked as a result of the event along with context about the event. Examples of typical event consumers include:

  - Event actuators, which are invoked to perform physical tasks such as operating valves, switches, or alarms

- Operator dashboards, which display information about behavior of IT systems and affected services
- Business dashboards, which display information about behavior of business processes
- Business processes, which can be initiated or resumed in response to an event
- Services and applications, which can be invoked in reaction to an event, and can include external content management systems or event repositories
- State machines, whose state can be changed in reaction to an event

This view of the conceptual architecture is based on the roles of each component, but that is not to say that a particular participant in the architecture cannot perform more than one role: an event producer could also carry out event processing and could act as an event consumer. In particular, the publication and subscription services are only required where a Publish/Subscribe-style model is used.

The conceptual architecture can be regarded as "nested," in that any participant could contain within it a network of further components. For example, an event producer might emit an event to the main event bus, but in the process of producing that event one could envision a *mini* version of the overall model, in which a producer emits an initial simple event for pattern matching with other events in a *mini* event bus, residing logically within the overall event producer.

Event bus components

The event bus transmits events from producers to consumers, and can provide additional services for processing and routing events. The event bus can have an associated event registry, and can have the capability to perform transactional storage of in-flight events (transient or persistent) by using an event repository.

The event bus can be local or implemented at an enterprise level, and the events received need to be processed based on the business requirement. This solution is achieved using simple and complex event processing services. These services are provided by event processing agents which are wired through event channels.

The services or building blocks that the event bus can provide are:

- Event channels, which transmit events from event emitters to the event bus, between components of the event bus, and on to event handlers
- Publication services, to enable producers to send events to appropriate channels.
- Subscription services, to enable dynamic registration of producers and consumers, such as allowing event handlers to find appropriate channels and subscribe to receive events from those channels
- Notification services to notify subscribed event handlers when events are available, supporting both push and pull of events
- Query services to allow a repository to be queried for events (and metadata)
- Event security services, to control access and authority for events; for example, to control authorization for adding and removing events to/from the event bus, as well as privacy and non-repudiation of event contents
- Event processing services, which provide filtering, transformation, and enrichment of events, and can also provide pattern matching and event derivation. This can include complex event processing, which processes events from multiple sources, and can carry out long-running pattern matching among events

- Event information services, which enable administrators to add, remove, and organize channels, to organize event type metadata (syntax and semantics) and to alternatively store event data in a relational format rather than using event message (that is, atomic) based persistence
- An event registry, to provide a taxonomy of event types and an ontology of event relationships
- An event repository, to store events for medium to long term event persistence

The most significant function types that must be provided by processing within the event bus are:

- Transformation: Function that transforms the incoming event by translating or splitting it
- Enrichment: Function that enriches the content of events with reference data from multiple possible sources
- Validation: Function to provide validation against required criteria
- Pattern detection: Function that recognizes actual and retrospective patterns; a combination from possibly multiple events, characterizing a significant business situation
- Filtering: Stateless function that filters events based on their content; that is, the information that is carried by the message generated when the event happened
- Aggregation: Function that can group events as necessary
- Routing: Function that routes events to the destination based on various possible routing patterns, such as pre-established itinerary, calendar-based, subscription or “intelligent” routing decisions

The conceptual architecture also includes event governance and security services, to manage and control the lifecycle of events and event metadata. Event monitoring and analytic infrastructure is needed for mainly administrative purposes, to notify failures in the event infrastructure, and to gather and display statistics about the event flow. These capabilities must cover the full event flow, and are therefore shown at the right side of the conceptual model diagram.

Thus the conceptual architecture represents event producers emitting events to the event bus, where they might be processed, and are finally consumed by event consumers. A consumer can as a result of an event produce another event, or react in some other means with another component, which itself produces an event as a result.

Figure 11 shows an example of how the conceptual architecture builds on the concept of an EPN, by illustrating components that are equivalents of EPAs, or sets of connected EPAs, and other components that provide event channel services.

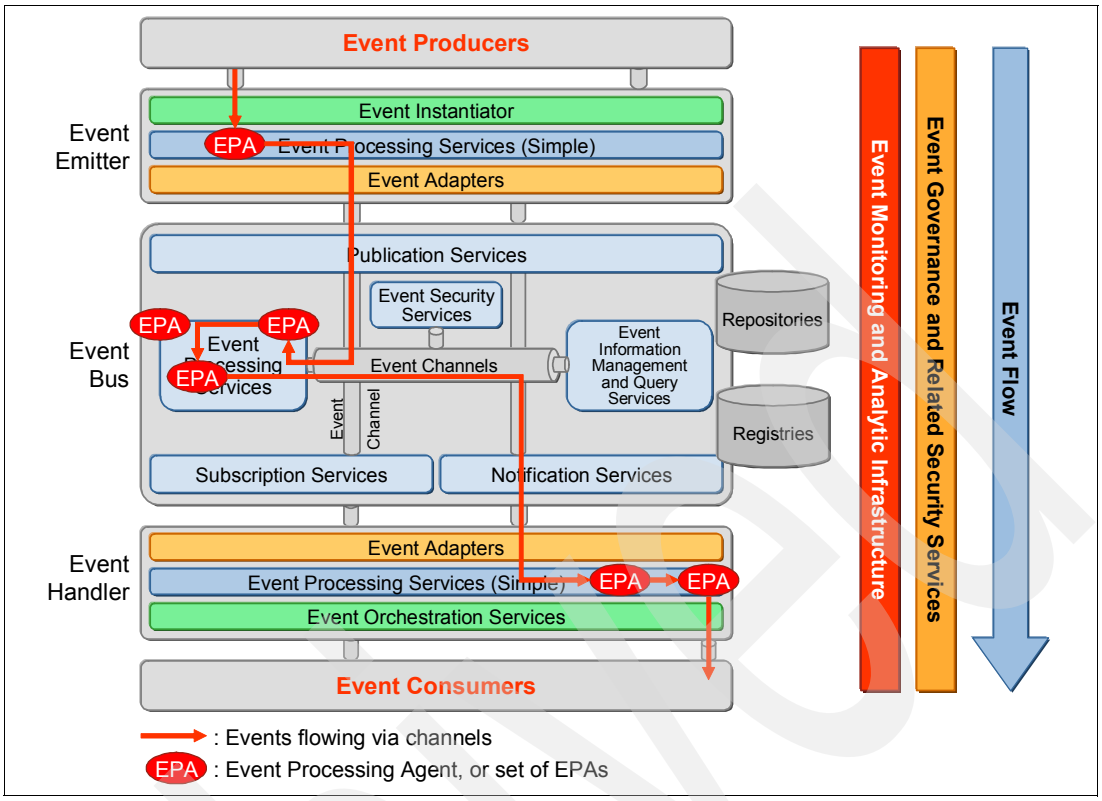

Figure 11 An example: EPN used by the Event Processing Conceptual Architecture

Flow of processing in the conceptual architecture

The purpose of this section is to describe the conceptual model in action. There are three phases to consider:

- Event emission phase
- Event processing phase
- Event consumption phase

This is not a single and consistent processing flow, and there is a very limited coupling between these three phases, particularly where complex event processing is involved; in that case, the three phases can occur in totally different time periods, and there is only a relation of cause and effect between the emitted event and the consumed event.

To provide concrete examples, this section refers to the scenarios that are described in sections “Event processing scenarios overview” on page 6, “Event Processing Network scenarios” on page 13 and “Mapping of the scenarios to the conceptual model” on page 32.

Event emission phase

The components of the conceptual architecture involved in this phase are mainly the event emitter’s components. Their role in the event emission can be quite technical; that is, they are mainly in charge of providing technical connectivity layers between the event producer and the event bus, but it can also be business oriented; that is, some business-oriented event processing logic or local EPAs can be implemented there.

Technical perspective

When a possibly meaningful business event occurs, the event producer sends an event message to the event bus using its local event emitter layer. The event instantiator sub-layer is a technical component in charge of detecting this specific business situation and of gathering all necessary information. The local event processing services can then process this information in order to create the event message, for example, by formatting it to comply with a generic format published in the event registry. This event message is then sent to the event bus by the event adapter sub-layer, which is in charge of adapting the event message to the transport protocols supported by the event bus.

Example: In the Fleet Management scenario we consider the delivery system, and particularly the vehicle dispatching subsystem as the event provider. Let us assume that the vehicle dispatching subsystem is a business application storing its data in a relational database. “Delivery change” is a business event that should be emitted each time the driver or vehicle in charge of a specific delivery is modified in this application. The event instantiator is implemented as a trigger on the table storing delivery data. This trigger is fired each time an instance of delivery is updated in this table. This trigger implements local event processing services. It validates that the update is related to a modification of the vehicle or the driver in charge of this delivery. If this test is successful, the trigger logic gathers all the necessary information (Driver1 ID, Vehicle1 ID, Driver2 ID, Vehicle2 ID, Sender ID, Receiver ID) and creates an instance of the “Delivery change” event message in a dedicated Delivery Change Event table. The event adapter sub-layer is implemented using a JDBC adapter, in charge of polling this table, and in charge of initiating external event processing logic at the event bus level.

Business perspective

In more advanced implementations, the event emitter layer can support more advanced producer-side event detection patterns and can implement local EPAs. Filtering, aggregation, enrichment, or even validation of elementary events can be considered in the local event processing sub-layer, in order to emit, over the event bus, a single event message characterizing the occurrence of a valuable business event.

Example: In the Public Health Alert Scenario, let us consider the “Hospital” as the event provider and the “Potential epidemic outbreak alert” event that it produces. Many patient situations are reported in the Hospital Information System. Among those, some of them need to be particularly monitored, because they are linked to a specific infection. The occurrence of such a situation is an elementary event. To detect a potential epidemic outbreak, it is necessary to match, locally in the “Hospital,” several similar elementary events, affecting many patients and over a limited time period. Therefore the event instantiator and the local event processing services implement an EPA in charge of:

- Validating the origin and quality of these elementary events
- Correlating them
- Producing a “Potential epidemic outbreak normalized” event as expected by the stakeholders of the EPN

Event processing phase

This phase takes place in the event bus. There can different flows of processing, involving different components of the event bus, depending on the number of events to be processed. Two behaviors are described here, for processing of a single event and for processing of multiple events.

Processing of a single incoming event

One possible basic processing pattern is publish/subscribe. When an event message is received at the event bus level, it is intercepted by the publication services sub-layer and

distributed to the various channels, which have been configured by the Event Bus Administrator. These channels are published in the event registry, to be made available for the event consumers or the subscription services component. Event consumers access the event registry to get the information about the channels associated with the event type they are interested in. Event consumers receive the event messages by directly listening to the relevant channels.

Example: In the Fleet Management scenario, consider the “Delivery change” event, with the Delivery Management Dashboard as an event consumer. Each time a “Delivery change” event is detected by the event bus, the related event message is made available on a specific event channel. The Delivery Management Dashboard is listening to this channel. When a new event message is received, the Delivery Management Dashboard processes it using its local event handler (see the section “Event consumption phase” on page 31).

It can also be the case that a single inbound event can generate multiple outbound events with different formats, depending on its payload and the subscription requests expressed for it. Event consumers can state their interest in a specific event type using the subscription services. They can also set additional parameters to specify the way they want to be notified of the related event. Among these parameters can be:

- Format of the event message to be received on notification
- Channel through which this message will be received
- Event filtering criteria

When an event message is received at the event bus, the publication services sub-layer processes each individual subscription request for this event type. Necessary event processing services are processed to mediate the event message (mainly filtering, enrichment and transformation services). Then notification services push the resulting event message to the event consumer through the requested channel.

Example: In the Public Health Alert scenario, let us consider the “General Practitioner” as the event consumer and the “Potential epidemic outbreak alert” event which it emits. The “General Practitioner” subscribes to health alert events because he wants to be notified through his e-mail each time a “Potential epidemic outbreak alert” event is raised in his specific area; he wants to get some additional information about the related disease. The event bus subscription services should offer some facilities that allow him to browse the list of available alerts, to position a filter on the geographic location, to choose the additional data that he might be interested in, and to select his preferred delivery channel. The event bus will use these parameters to filter incoming “Potential epidemic outbreak alert” events, to enrich them with the requested additional information, and to format it as an e-mail to be pushed through notification services to the consumer.

Processing of multiple incoming events

In this case, multiple incoming events are considered, possibly spanning different types, possibly emitted from multiple sources over a specified time period, referred to here as an event set. A meaningful business situation is not detected at the event producer level, but within the event bus, by matching multiple events from the event set. The event bus implements one or many EPAs in charge of carrying out this detection. One or many patterns will have been set by the event bus administrator (potentially using the subscription services, or event information management services) and stored in the event registry. These patterns can encompass multiple dimensions including causality, time, location or many others. When an event is received, the event query services can be used to check in the event registry whether it belongs to an event set associated with valid pattern matching rules. If so, the event processing services sub-layer is in charge of applying this rule to the received event.

The event bus in this case can already be waiting for an event of this type because it has already received at least one event for this event set, or a new processing flow could be initiated because no event for this event set has been received so far.

The processing of this pattern matching rule produces a resulting event, which can be:

- Directly published in the relevant channels
- Mediated and transmitted through the relevant channels to the interested subscribers
- Transmitted to another EPA in charge of another processing rule, with which this event type is associated

This event can be recursive behavior, and the chained processing of pattern matching makes up an event processing flow. The time period is often a significant characteristic of the event set and time is also an important factor in considering implementation of the event processing services in the event bus.

The event repository can play an important role in event processing, and especially for complex event processing. It keeps track of events processed in event flows, either incoming events, generated events, or resulting events.

- This is very useful for monitoring purposes, for example, to answer such key questions as: which combination of incoming events or which sequence of event processing agents produced this result?
- Because an event set can be considered over a potentially long time period, it can be necessary to persist related events.

Example: In the Fleet Management scenario, consider the “Delivery driver exceeded driving time” agent and the associated events: “Driver starts working” and “Driver ends working.” These two event types and a time period of more than 8 hours characterize the event set. Each time a “Driver starts working” event is received at the event bus level for a specific Driver ID, an EPA should be initiated to wait for a “Driver ends working” event for the same Driver ID. If this event is more than eight hours later than the initial one, a resulting event “Driver exceeded driving time” event should be issued.

Example: In the Public Health Alert scenario, consider the “Potential epidemic outbreak alert” event and, as the event consumer, the “Match potential epidemic outbreak alerts” agent. The “Potential epidemic outbreak alert” event type and a time period of 2 weeks characterize the event set. Each time an event message of this type is received by the event bus, it is transferred to an EPA implementing the agent pattern matching logic: if a “Potential epidemic outbreak alert” is received from ten different sources within less than two weeks, a “Pandemic outbreak alert” should be raised.

Event consumption phase

This phase takes place mainly in the event handler layer associated with the event consumer. As with the event emission phase, this layer can play either a technical role, or a more business-oriented one.

Technical perspective

In this phase, the event consumer receives an event message from the event bus and processes it, relying on its local event handler layer. The event adapter sub-layer is in charge of receiving the event message and interfacing with the transport protocols supported by the event bus. There can be a preliminary processing of the event message, implemented at the local event processing sub-layer. This processing can be, for example a technical adaptation of the event message payload in order to comply with event-consumer-specific input formats. The event orchestration sub-layer identifies the piece of business logic implementing the action associated with the reception of this specific event message. The event orchestration

sub-layer initiates the execution of this logic, with the optionally transformed payload of the event message as input parameter.

Example: In the Fleet Management scenario, consider the “Delivery change” event, and the Delivery Management Dashboard as its event consumer. The Delivery Management Dashboard is implemented as a portal. A new “Delivery change” event message is received at the local event handler via a WebSphere® MQ message queue connection (local event adapter), which is the interface with the event bus. The local event processing logic is implemented as a portlet which gets the event payload, and processes it to update some indicators and to modify graphics that are presented to the end user.

Business perspective

The event handler layer can behave, in more advanced implementations, as a convergence point of multiple elementary event messages. These elementary event messages are matched in the local event processing sub-layer. This produces a local resulting event, associated with an action at the event consumer. The resulting event is then transmitted to the event orchestration layer to initiate the relevant business logic in the event consumer. On the other hand, reception of a single event message can generate multiple local processing because many parties at the event consumer side are interested in it in different ways.

Example: In the Public Health Alert scenario, let us consider the “Hospital” as the event consumer and the “Pandemic Outbreak Alert” the event that it consumes. When such an event is notified to the “Hospital,” then as it raises a highly critical situation, there can be several strands of local processing of this event in parallel:

- The information can be directly pushed on the Hospital Intranet Portal through a news channel.
- Some key professionals can be recalled via an SMS message.
- The event message can be sent to logistics applications in order to initiate specific processes in charge of managing urgent health situations.

Mapping of the scenarios to the conceptual model

In this section, we revisit our event processing scenarios, and show how they map onto components of the conceptual architecture.

Fleet Management scenario mapping

Having defined the various actors (event producers and event consumers), events and event processing agents in the section “Fleet Management scenario details” on page 13, a mapping of the Fleet Management scenario to the event processing conceptual architecture is now possible (see Figure 12). The linkage to the event producers and event consumers is quite obvious. All agents except for A4, the routing agent, are mapped to the event processing component in the event bus. There is no mapping to the event emitter nor to the event handler part of the conceptual model because the two events, E1 and E3, produced by the Driver Report System, require no special treatment and can be processed by agents A1 and A2 as is, and E5 can be consumed by all of the consumers as is.

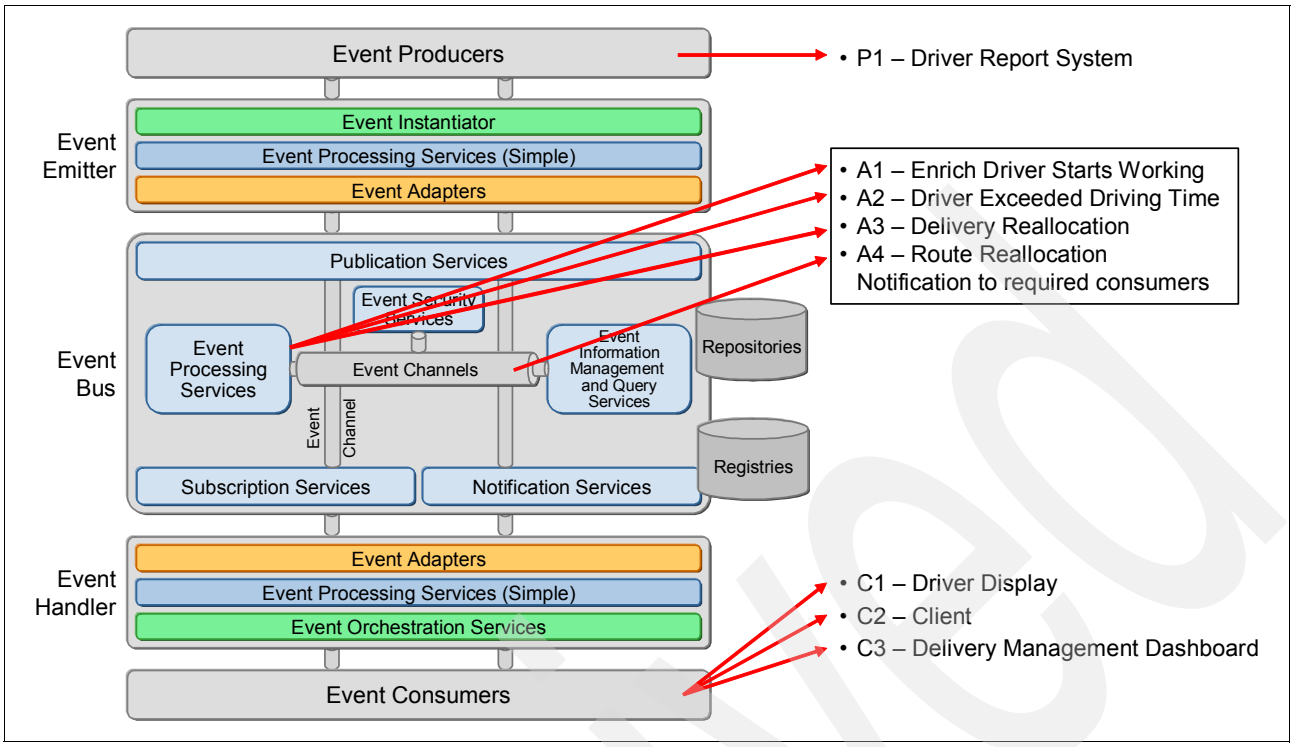

Figure 12 Conceptual Architecture components used in Fleet Management Scenario

Event processing is important in the Fleet Management scenario because it allows all the disparate information regarding the drivers' locations, driving times, and delivery routes to be responded to in a timely way. It also offers scope to easily change the rules regarding permitted driving times, how delivery changes are handled, and so on.

Public Health Alert scenario

After defining the various actors (event producers and event consumers), events and event processing agents, a mapping of the Public Health Alert scenario to the event processing conceptual architecture is also now possible (see Figure 13). The linkage to the event producers and event consumers is obvious. There is some mapping of event processing agents to the event emitter, and the main processing is linked to specific components in the event bus. There is no mapping to the event handler part of the conceptual model because this is not required by the aspects of the scenario that have been detailed here.

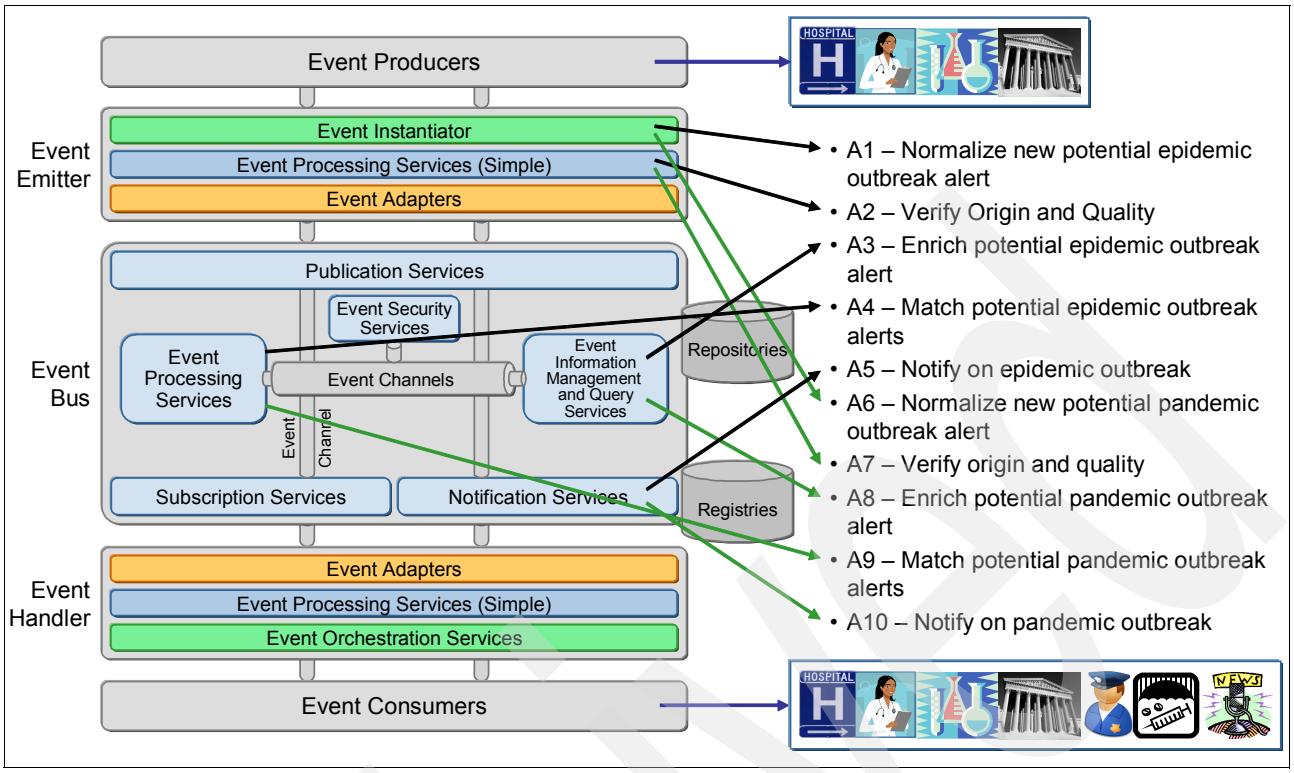

Figure 13 Conceptual Architecture components used in Public Health Alert Scenario

Having completed describing the scenario, let us consider why event processing is important for the Public Health Alert example.

Because the scenario is related to public health, time is an important factor, as are event history and the number of event producers and event consumers. Three main factors are important to understand and are effectively defined in the event processing system:

- Loose coupling between event producer and event consumer – the origin of the event is not important (as long as quality and source can be identified by the event consumer).
- Starting event processing as soon as the event has been detected and providing notification as soon as an epidemic or pandemic outbreak has been identified is a very important factor.
- Because the event processing system is not dependent on the business processes at the event consumer side, this allows for a highly flexible and dynamic event processing network that provides value to a large number of possible event consumers.

Communication Service Provider scenario

Figure 14 shows the mapping of the Communication Service Provider scenario to the components of the Event Processing Conceptual Architecture.

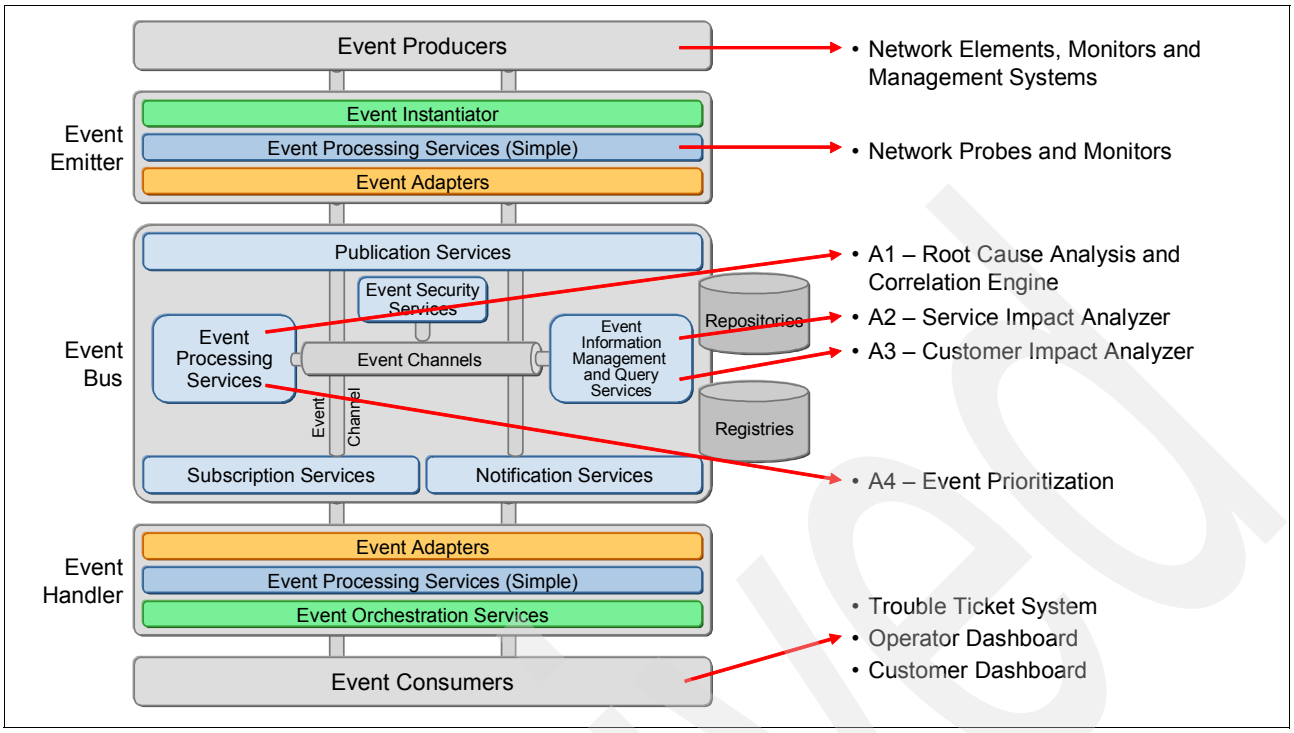

Figure 14 Conceptual architecture components used in Communication Service Provider scenario

Conclusions

We have introduced a Conceptual Model for Business Event Processing, consisting of a conceptual architecture built upon the concept of an Event Processing Network. This conceptual architecture shows how events generated by event producers can be prepared and processed for consumption by event consumers, with an intermediate event bus which provides services enabling events to be filtered, enriched, formatted, routed, aggregated or split, and so on as required. This conceptual architecture drills down into capabilities which can be required at the producer and consumer side, such as some simple event processing capabilities and event adapters. The conceptual architecture also indicates the various services which can be required in the event bus. The intention of this conceptual architecture is to identify all components which might be required in realizing any event processing implementation, but only a selection of components is expected to be needed for any particular implementation or scenario.

The paper introduces a number of scenarios which illustrate the use of event-driven processing to provide business value, and maps these to the conceptual model (event processing network and conceptual architecture) to show how this conceptual model can apply in a selection of practical situations. We have given a description of each scenario, explained why event processing is important for the scenario, enumerated the producers, consumers, events and event processing agents required to realize the scenario, and demonstrated the mapping to the conceptual architecture.

As the business value and opportunities provided by business event processing become more widely recognized, it will be important to have a conceptual model and architecture on which to base the logical and physical architectures that will be used to implement business event processing solutions.

Other resources for more information

For more information, consult the following resources:

- "The Growing Role of Events in Enterprise Applications", Gartner Group, W.Roy Schulte, AV-20-3900 (2003) available at Web address:

  http://www.gartner.com/DisplayDocument?doc_cd=116129

- IBM Solution brief "Fleet optimization for travel and transportation", IBM, 2009, TTS03009-USEN-00,

  ftp://ftp.software.ibm.com/common/ssi/pm/sp/n/tts03009usen/TTS03009USEN.PDF

- IBM Redbooks publication, *Healthcare Collaborative Network Solution Planning and Implementation*, SG24-6779

- *Event Processing Network and Implementation*, Etzion, O. and G. Sharon, IBM Systems Journal, Vol. 7 No. 42 (2008).

- The Event Processing Technical Society (EPTS) has published a glossary of event processing terms, available from the Document Management section of their public site, available at this address:

  http://www.ep-ts.com/

The team who wrote this guide

This guide was produced by a team of specialists from around the world.

Catherine Moxey is an IBM Senior Technical Staff Member in the CICS® Transaction Server for z/OS® team, based in Hursley, UK, and is the architect for event processing support in CICS. She is a member of the Event Processing Technical Society and active in its Reference Architecture workgroup.

Mike Edwards is a Strategist in Emerging Technologies based at the Hursley Lab in the UK. He is a co-chair of the OASIS SCA Assembly technical committee and on the Apache Tuscany project. He is also a contributor to the specifications for the European PEPPOL project.

Opher Etzion is an IBM Senior Technical Staff Member and Master Inventor. He is the Event Processing Scientific Leader in the IBM Research Lab in Haifa, Israel, and is Chair of the Event Process Technical Society (EPTS) Steering Committee.

Mamdouh Ibrahim is an IBM Distinguished Engineer and CTO of the Enterprise Architecture and Technology practice. Dr. Ibrahim's responsibilities include development of EA and SOA assets and providing architecture expertise and consulting to clients. Dr. Ibrahim holds two Bachelor degrees in Electrical Engineering and Mathematics; three Master degrees in Mathematics, Solid® State Science, and Computer Science, and a Ph.D. in Computer Science. He is a member of IEEE, ACM, and is an adjunct faculty member at Central Michigan.

Sreekanth Iyer is an Senior IT Architect with the IBM India Software Lab Services and Solutions team building IBM SOA solutions for customers and partners.

Hubert Lalanne is an IBM Distinguished Engineer and Software Information Technology Architect (SWITA), based in France. He is also a member of the IBM Software Group WW Technical Leadership Council and IBM Technical Expert Council France & NW Africa.

Mweene Monze is Executive IT Architect in IBM Software Group, based in Johannesburg, South Africa, with responsibility for Public Sector Technical Sales.

Marc Peters is a Senior Software IT Architect for Energy & Utility Customers based in Cologne, Germany, leading opportunities and projects in SOA, Event Processing, and IoD linked to E&U industry business requirements. Marc has more than 17 years of experience in IT in international projects. He is a frequent speaker at internal and customer events.

Yuri Rabinovich is researcher in the Event-based Middleware and Solutions group in the IBM Haifa Research Lab, Israel. He received his M.Sc in Information Systems Management at the Technion, the Israel Institute of Technology. He joined IBM Haifa Research Lab in 2006 focusing on development of expressive Complex Event Processing rule engine AMiT (Active Middleware Technology). He led Scalable and Distributed Event Processing research projects and developed new techniques that improve event processing performance targeting at WebSphere Business Events engine.

Guy Sharon manages the Event-based Middleware and Solutions group in the Haifa Research Lab, Israel. He received his M.Sc in Information Systems Management at the Technion, the Israel Institute of Technology, where his thesis was on the conceptual model of Event Processing Networks. He joined IBM Haifa Research Lab in 2000 focusing on Active Technologies and was part of the AMIT (Active Middleware Technology) R&D team.

Kristian Stewart is an architect for Tivoli® Network Availability Management.

The authors would also like to acknowledge the valuable contributions to the conceptual model and to this paper made by:

- Christopher Ahrendt
- Kyle Brown
- Koteswara R Chejarla
- Norman Cohen
- John Dinger
- Greg Flurry
- Paul Giangarra
- Kevin Hall
- Robert Heuchert
- Beth Hutchison
- David H Janson
- Jojo Joseph
- Chung-Sheng Li
- Rahul Narain
- Peter Niblett
- Dave Russell
- Robert Sawyer
- Boris Shulman
- Michael Spicer
- Bobby Woolf

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author - all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in

length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/pages/IBM-Redbooks/178023492563?ref=ts

- Follow us on twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

This document, REDP-4642-00, was created or updated on March 23, 2010.

Trademarks