Event-Driven SOA

by Jürgen Kress, Oracle, Hajo Normann, Oracle ACE Director, Danilo Schmiedel, Senior Consultant, Optiz Consulting, Guido Schmutz, Technology Manager, Trivadis, Bernd Trops, Senior Principal Consultant, Talend Inc., Clemens Utschig-Utschig, Chief Architect, Shared Service Centre, Global Business Services, Boehringer Ingelheim, Torsten Winterberg, Business Developement and Innovation, Optiz Consulting, and Berthold Maier, Enterprise Architect, T-Systems International department of Telekom Germany

Abstract: If we consider real companies and their business transactions, we see that the real world is not really service-driven at all, but much more event-driven. A new customer is created in the system, a new car reservation is made, a vehicle is returned or needs to be taken to the shop. All of these "functions" can be supported by services, but often also by precisely defined process chains.

However, complex business processes can rarely be automated "in one piece," as the real exceptions and dependencies of diverse business processes are highly dynamic. That brings us to the point where the concept of "event" becomes useful in our architectures. In this article, we therefore want to shine light on event-driven architectures and tie them into our current argumentation chains in SOA.

Dealing with Business Events

Most companies now collect business-relevant information by aggregating available data. As a rule, this is in the domain of data warehouses which condense information a follow-up to preceding events. Here the focus is directly on the data and not on the process information, which is actually much more interesting. We miss out on the ability to process business information in near-realtime, which would allow the company management to react much more quickly to events as they occur. The discipline of business intelligence is developing rapidly in order to take this issue into account. In the following, we want to consider the underlying mechanisms, or the events which allow faster reactions to important changes.

What is an Event?

The term "event" generally has two meanings. On one hand it denotes a perceptible occurrence, and on the other, its representation in a computer system. The latter is also called an "event object." The term itself is therefore ambiguous. What interests us is the "business event," namely the status change in a company.

What does that mean exactly? As shown in Figure 1, an event is a type of "thing which happens," which, however, for us has no meaning as long as we are not informed of this in the form of a representation in our IT systems. The occurrence of an event in our view therefore always goes along with a notification. As a whole, we use the term "event occurrence." This means the occurrence of an event, including the notification that this event has occurred.

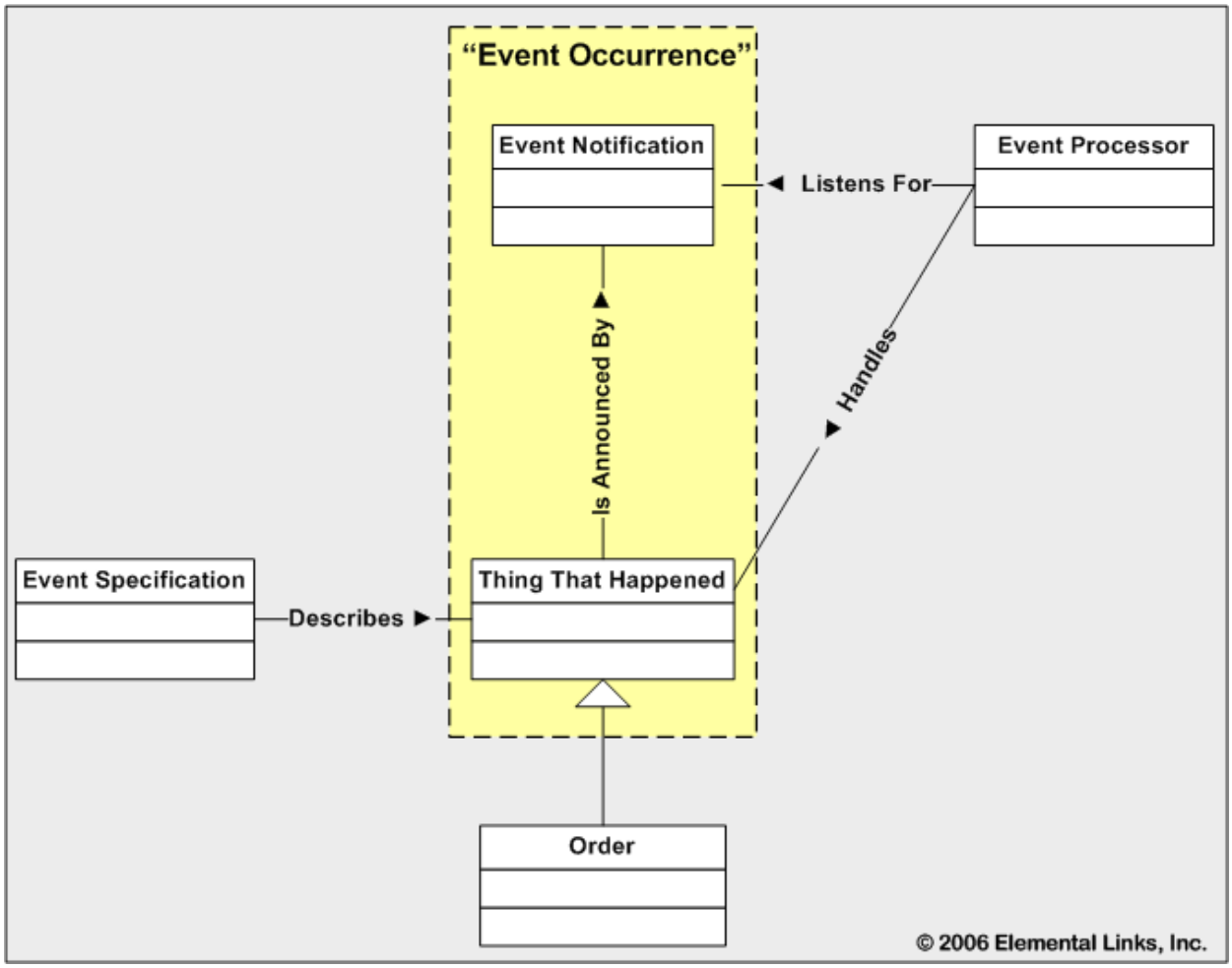

Figure 1 - Event and event notification.

The "event occurrence" is described by a specification, which is used to create a type of class in the Java sense. Specific characteristics of this class, or characteristics of this "event occurrence," can be, for example, individual events with an order, which are processed by an "event processor." A unique identifier for event referencing, a timestamp of the creation, the event source, and the event type are some of the components that belong to the minimal description of an event in the form of metadata.

The terms "event class," "event definition," or "event diagram" can be used interchangeably for the event type. It is important that all events must be instances of a certain event type, as the metadata structure is determined in this way and represented by a collection of attributes and properties. The "event processors" can therefore react differently to events of different types.

Listing 1 shows an example of the description of a technical event in the form of an XML message, which can also be set up as a POJO or a similar type of object:

```
<event>
  - <eventHeader>
    <eventSpecificationId>1</eventSpecificationId>
    <eventType>Low Inventory Threshold</eventType>
    <eventClass>Business</eventClass>
    <eventSubClass>Threshold</eventSubClass>
    <eventOccurenceId>4019</eventOccurendeId>
    <eventTimeStamp>2006-26-06 13:42:00:01</eventTimeStamp>
     <eventSource>Inventory Monitor</eventSource>
  </eventHeader>
  - <eventBody>
    <sku>12345678</sku>
    <productName>A Good Book</productName>
    <currentInventory>12</currentInventory>
    <lowThreshold>15</lowThreshold>
    <highThreshold>75</highThreshold>
    <reorderAmount>42</reorderAmount>
  </eventBody>
</event>
```

Listing 1 – A technical event's description is shown as an XML message.

In many connections, therefore, events are very similar to traditional messages, which are exchanged via MoMs. There are no large payloads and the content can be expanded by the event processors if necessary. We will discuss some differences in the approach below.

What is EDA?

"Event-Driven Architecture" is a system architecture which, in the simplest case, executes and manages rules in the following form:

If the reality differs from the expectation, then update the expectation and provide an answer.

If we have put particular value on loose coupling in an SOA, EDA goes one step further: As an event is generated neutrally from an event source and sent to a middleware to be processed, the functionality that is triggered by the event is no longer known. In an SOA, the specific service call must at least have been activated. In an EDA, we tend to use the term "decoupling" rather than loose coupling.

An event-driven application consumes, processes, and generates events as described in a very loosely coupled, or decoupled, way. An EDA supports “event-driven applications.” The processing middleware accepts the events, assesses the content, checks against a criteria catalog if necessary, and then informs interested consumers via publishing or subscription. The consumers process the event which, like the event source, can even be implemented as a service.

Event Processing

Event processing is the performance of operations on events, such as reading, changing, creating, and deleting. A prerequisite for EDA is that the reality may differ from the expectation. Therefore, both the expectation and the significant difference, as well as the corresponding response behavior, should be specified. Figure 2 shows a typical processing chain for event processing:

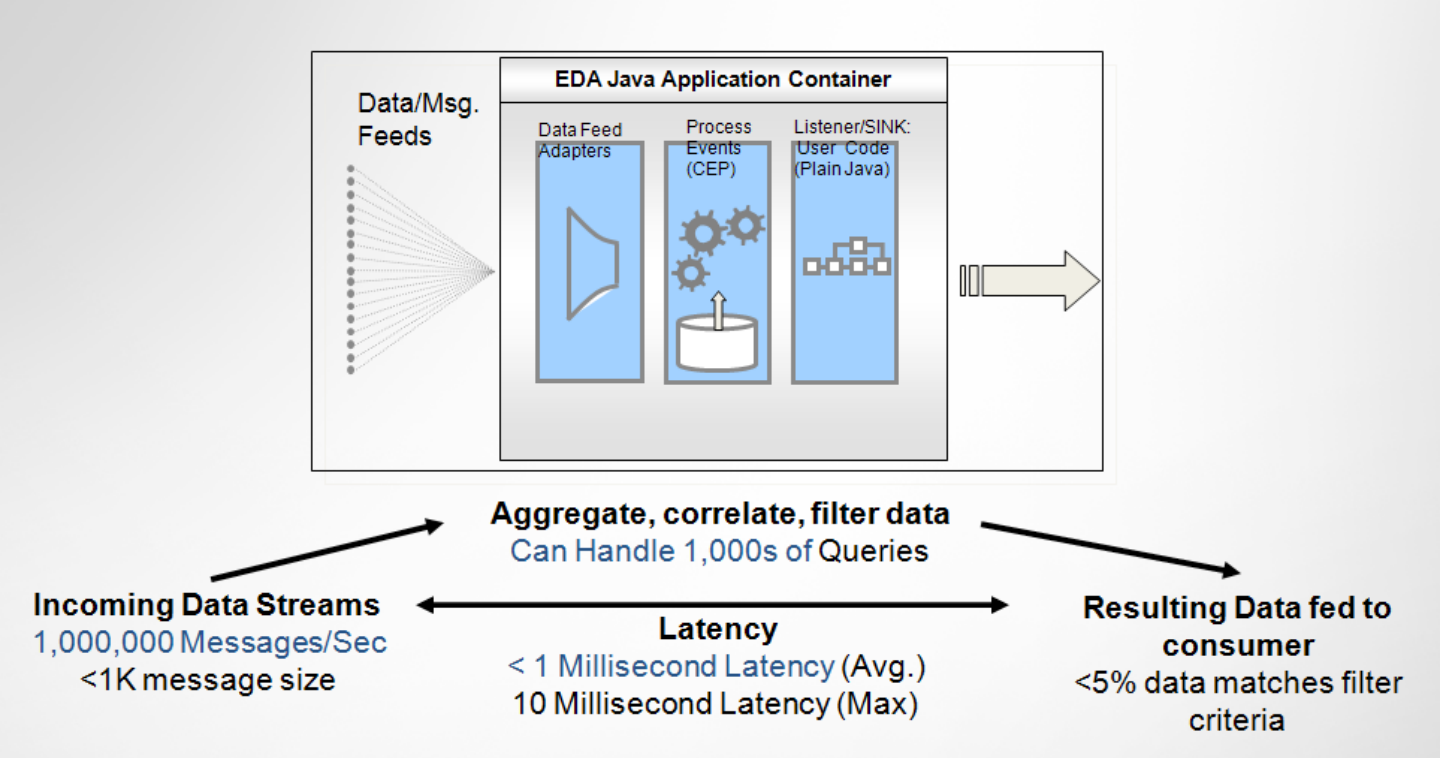

Figure 2 - The event sources (generators) create events and position them in an event channel (notification). An event processing engine processes the events in the event channel and triggers corresponding actions.

An acute problem in our current complex architectures is that business events are often only insufficiently perceived and adequately handled. Individual events are not meaningful and leave important business connections out of consideration. Individual events occur too frequently for the causes and effects of each event to be analyzed and treated adequately. Filtering the relevant events reduces their number, but does not clarify their meaning. This means that after performing the “event detection,” we have to move to the “event correlation” phase.

The results must be put in relation to each other, as the business significance of event correlations is considerably higher than that of individual events. In addition, the number of events which need further processing is also reduced due to the event correlations. The challenge of this phase is that event correlations are often hard to identify, and the events are usually distributed across the whole application landscape. Recognizing patterns within the mass of events is therefore of particular significance.

The third phase is the reaction to the events. Reacting to simple events is not problematic, like using service calls or status messages. There are many infrastructures available for this, such as SOA or BAM dashboards. However, the status information is, at least currently, interpreted in the user's head most of the time. There is no automatic comparison with complex event patterns or with combinations of events from vastly different sources. The current reaction potential is therefore very dependent on the employees, their experience, and acute attention span. Complex Event Processing engines can provide support here.

The discipline of Complex Event Processing (CEP) therefore clearly differs from the traditional processing mechanisms based on queues or databases. Temporary storage is necessary, which allows complex correlations of different types of events, even over long time horizons. This is also called "in-stream processing," which is becoming increasingly important, as shown in the appearance of SQL-type, stream-based query languages.

Simple, Stream, and Complex

We generally differentiate between three types of event processing.

In Simple Event Processing (SEP), only "relevant" events are put in the event channel for processing. Consider the examples where "car class XY is sold out" or "warehouse stock of winter wheels running low." This is very similar to the message-based systems. The processing logic assesses event type and content and then reacts accordingly.

In addition to the "relevant" events, in Event Stream Processing (ESP) the "usual" events, such as all general orders or RFID events, are also published in the event channel. This puts higher demands on event processing, but also allows more options for evaluation.

Finally, Complex Event Processing (CEP) deals with the co-occurrence of different events in order to derive actions from them. The events in the event channel can have different event types and occur over long periods of time. The event correlation can be in terms of content, time, or location. CEP is therefore often used for detecting anomalies, risks, or even opportunities in the business, with the clear aim of benefiting from the advantages of the more up-to-date information compared with traditional reporting solutions and ultimately gaining the upper hand over the competition.

What exactly is the difference between ESP and CSP? ESP is the processing of streams, where an event stream is a chronological sequence of events, such as a stock ticker. CEP, however, works on "clouds" that are known as "event clouds." An event cloud is the result of several event-generating activities from different parts of an IT system. An event cloud could contain several streams. A stream is therefore a special instance of a cloud.

Using the chronological order within a stream has its advantages: processing is fast, as only a few events need to be held in the buffer. However, dependencies are important in clouds: Which dependent events have occurred? Or often even more exciting: Which events perhaps did not occur?

This makes it clear that Event Stream Processing is designed more for high-speed processing, whereas CEP focuses on the extraction of information from the event clouds. In practice, the differences between ESP and CEP become blurred, so the more powerful CEP dominates.

Event Server Infrastructure

There are no big surprises in the required infrastructure for event servers. Enterprise service buses or messaging infrastructures are used for transporting and routing events. Complex Event Processing engines are used for filtering, aggregating, and correlating events, which can often be docked as a JBI service engine on an ESB. The end users are integrated through Business Activity Monitoring (BAM) dashboards, or separate service or process calls.

Figure 3 shows a typical infrastructure:

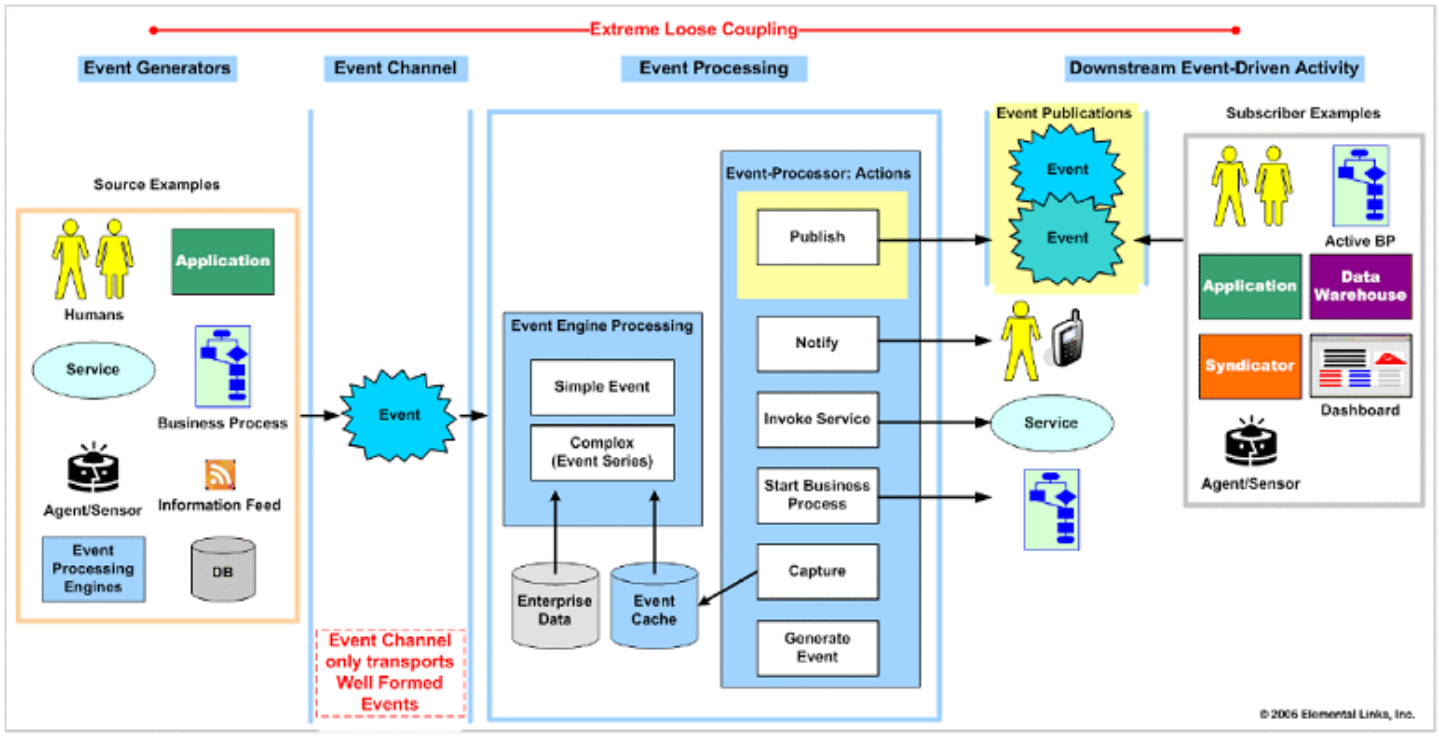

Figure 3 - Event server infrastructure.

SOA vs. EDA

As already established in the introduction, the world is rather more event-driven than service-driven. SOA identifiers are the loose coupling of the services, 1:1 client-initiated communication between the provider and consumer, and synchronous response behavior, for the most part. Identifiers of EDA are decoupled interactions, n:m communication, event-driven actions, and asynchronous operations. From our point of view, it is not necessary to decide on one side or the other. SOA provides a very solid basis for EDA and applications can use both styles. A component should use SOA for a service call if:

- it is known exactly which service is to be called
- the service will be called just once
- an answer about the completion of a service is expected
- an answer is expected

A component should use EDA for an event publication if:

- all receivers are to be informed if they are interested in it, in certain circumstances
- it is not known which receivers are interested in the event
- it is not known how receivers react to this event
- different receivers react differently to the same event
- it concerns a one-way communication from the sender to the receiver

In this respect, we can apply the equation "2 + 2 > 4," as the combination of both architecture styles provides more than the sum of their individual parts. SOA uses the request-reply communication pattern (possibly asynchronously, with long time intervals between request and response) when executing pre-defined processes and logic. In comparison, ED applications use the typical publisher/subscriber pattern, processing large amounts of events in certain circumstances with the aim of creating fewer, new "actionable" events. SOA and EDA are complementary: together they allow the creation of on-demand applications with high business value.

An Eventful Flight

Anyone who frequently travels by plane is intimately familiar with the unpleasant question of "Where's my luggage?" At the check-in desk, the passenger and his luggage are separated from each other. A range of processes is necessary to reunite the two at the end of the flight, including:

- check-in counter
- security
- baggage handling
- gate operations
- flight operations
- airport operations control (BAM) dashboard
- customer service

The optimum interim result would be if the plane takes off on time with the passenger and luggage on board. However, as many of us know, there are numerous possible events which can complicate matters. The luggage may get lost between check-in and loading. The passenger may be late due to the line at security. The luggage may contain material that is not allowed to be carried and need to be searched. The flight may be canceled and the passenger may check in somewhere else. The passenger may decide to change travel plans after check-in. Many other complications are possible.

Can SOA, EDA, or a combination of both provide optimum IT support in this scenario?

An important insight is required for making the decision: the scenario depicted does not represent a single "boarding service" or process. It is a range of services/processes which interact with each other. The interaction is complex, depending on several boundary conditions, and is therefore a typical scenario for "sense and respond" or EDA technology, which initiates services of an SOA if necessary. Attempting to unite such a scenario in a single executable process will inevitably end up in chaos.

Process	Event
Check-In	Passenger checked in, bag(s) checked
Security	Passenger enters/exits security
Baggage Handling	Bag scanned at checkpoint, bag loaded into container
Gate Operations	Flight open, boarding, final boarding, closed
Flight Operations	Flight at gate, containers loaded, departure, take-off
Customer Service	Bag rechecked on new flight

Table 1 – The various events of a flight.

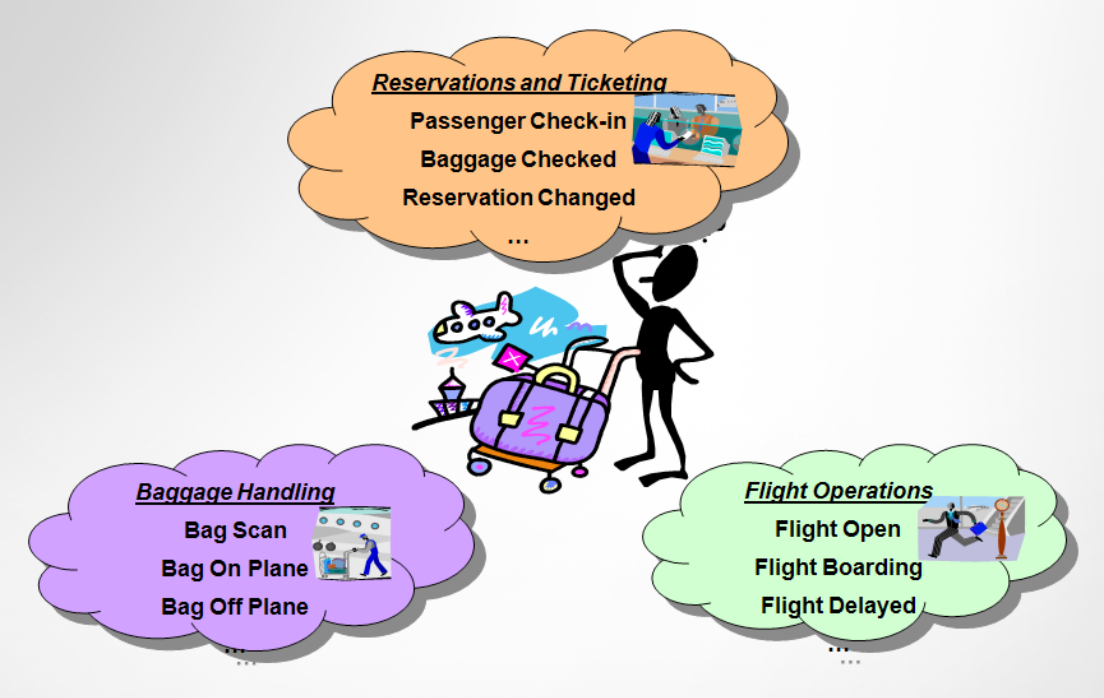

Figure 4 – Flights: A world of events.

Events and Distributed Data Management

A very simple but effective field of application for events is to be the solution of problems concerning distributed data management. If it is not possible to create a single data access point, a Business Entity Service which brings together everything to do with the rental cars using events according to Simple Event Processing, for example, is a viable solution. When entering the data in the database, a “car_rented-event” can be created which is picked up by simple processing logic. The necessary information can be permanently stored in further systems in this way.

Conclusion

We have shown that SOA and EDA are not competing architectures but rather complement each other very well. EDA often provides a good depiction of the real world, and the events can often be easily read from the process models. EDA is more efficient than SOA if there are several consumers for certain data, since not many service calls need to be programmed. This is because the unique generation of an event in the source system is sufficient. The availability of powerful middlewares with special EDA features has recently increased. The inexperience of application developers in designing event-driven systems and the increased challenges in troubleshooting due to the significant decoupling of the individual components are currently on the negative side. However, there are also initial signs in the suites of large manufacturers that this complexity will be addressed with improved runtime monitoring.

References

[REF-1] Berthold Maier, Hajo Normann, Bernd Trops, Clemens Utschig-Utschig, Torsten Winterberg: “Rent your Car – Service-Oriented,” Java Magazine 11/2008

Jürgen Kress

An eleven year Oracle veteran, Jürgen works at EMEA Alliances and Channels. As the founder of the Oracle WebLogic and SOA Partner Community and the global Partner Advisory Councils. With more than 4,000 members from all over the world the SOA and WebLogic Partner Communities are the most successful and active communities at Oracle. Jürgen hosts the communities with monthly newsletters, webcasts and conferences. He hosts his Fusion Middleware Partner Community Forums, where more than 200 partners get the latest product updates, roadmap insights and hands-on trainings. Supplemented by many web 2.0 tools like twitter, discussion forums, online communities, blogs and wikis. For the SOA & Cloud Symposium by Thomas Erl Jürgen was a member of the steering board. He is also a frequent speaker at conferences like the SOA & BPM Integration Days, Oracle Open World or the JAX.

Contributions

- Event-Driven SOA
- SOA in Real Life: Mobile Solutions
- SOA and User-Interfaces
- Understanding Service Compensation
- Securing the SOA Landscape
- Enterprise Service Bus
- Canonizing a Language for Architecture: An SOA Service Category Matrix
- Industrial SOA
- SOA Blueprint: A Toolbox for Architects

Berthold Maier

Berthold Maier works in the T-Systems International department of Telekom Germany as Enterprise Architect. He has more than 19 years experience as developer, coach and architect in the area of building complex mission critical applications and integrations scenarios. Within eleven years as Oracle employee he has held several leading positions including chief architect in the consulting organization. Hi is the founder of many frameworks and take over the responsible for reference architectures around BPM/SOA and Enterprise Architecture Management. Berthold is also well-known as a conference speaker, book author and magazine writer.

Contributions

- Event-Driven SOA
- SOA in Real Life: Mobile Solutions
- SOA and User-Interfaces
- Understanding Service Compensation
- Securing the SOA Landscape
- Enterprise Service Bus
- Canonizing a Language for Architecture: An SOA Service Category Matrix
- Industrial SOA
- SOA Blueprint: A Toolbox for Architects

Hajo Normann

Hajo Normann works for Accenture in the role of SOA & BPM Community of Practice Lead in ASG. Hajo is responsible for the architecture and solution design of SOA/BPM projects, mostly acting as the interface between business and the IT sides. He enjoys tackling organizational and technical challenges and motivates solutions in customer workshops, conferences, and publications. Hajo leads together with Torsten Winterberg the DOAG SIG Middleware and is an Oracle ACE Director and an active member of a global network within Accenture, as well as in regular contact with SOA/BPM architects from around the world.

Contributions

- Event-Driven SOA
- SOA in Real Life: Mobile Solutions
- SOA and User-Interfaces
- Understanding Service Compensation
- Securing the SOA Landscape
- Enterprise Service Bus
- Canonizing a Language for Architecture: An SOA Service Category Matrix
- Industrial SOA
- SOA Blueprint: A Toolbox for Architects

Danilo Schmiedel

Danilo Schmiedel is one of the leading BPM and SOA System Architects at OPITZ CONSULTING. He has been involved in large integration-, business processes automation and BPM / SOA development projects where he implemented solutions for various customers. His main field of interest is focused on the practical use of BPM and SOA on a large scale. Additionally he works as BPM and SOA project coach. Danilo is a frequent speaker in the German Java and Oracle communities and has written numerous articles about the above topics. Before joining OPITZ CONSULTING Danilo worked as Software Engineer in several international projects. The Leipzig University of Applied Science has awarded his outstanding reputation in 2009.

Contributions

- Event-Driven SOA
- SOA in Real Life: Mobile Solutions
- SOA and User-Interfaces
- Understanding Service Compensation
- Securing the SOA Landscape
- Enterprise Service Bus
- Canonizing a Language for Architecture: An SOA Service Category Matrix
- Industrial SOA
- SOA Blueprint: A Toolbox for Architects

Guido Schmutz

Guido Schmutz works as Technology Manager for the IT services company Trivadis. He has over 25 years as a software developer, consultant, architect, trainer, and coach. In Trivadis he is responsible for SOA, BPM and application integration, and is head of the Trivadis Architecture Board. His interests lie in the architecture, design, and implementation of advanced software solutions. He specializes in Java EE, Spring, Oracle SOA Suite and Oracle Service Bus. He is a regular speaker at international conferences and is the author of articles and several books. Guido is an Oracle ACE Director for Fusion Middleware & SOA.

Contributions

- Event-Driven SOA
- SOA in Real Life: Mobile Solutions
- SOA and User-Interfaces
- Understanding Service Compensation
- Securing the SOA Landscape
- Enterprise Service Bus
- Canonizing a Language for Architecture: An SOA Service Category Matrix
- Industrial SOA
- SOA Blueprint: A Toolbox for Architects

Bernd Trops

Bernd Trops is a Senior Principal Consultant at Talend Inc. In this role he is responsible for client project management and training.

Bernd is responsible for all Talend projects within the Deutsche Post and the introductions of new versions and components.

Before Talend, Bernd was a Systems Engineer working on various projects for GemStone, Brocade and WebGain and therefore has extensive experience in J2EE and SOA. From 2003 to 2007 Bernd Trops worked as a SOA Architect at Oracle.

Contributions

- Event-Driven SOA
- SOA in Real Life: Mobile Solutions
- SOA and User-Interfaces
- Understanding Service Compensation
- Securing the SOA Landscape
- Enterprise Service Bus
- Canonizing a Language for Architecture: An SOA Service Category Matrix
- Industrial SOA
- SOA Blueprint: A Toolbox for Architects

Clemens Utschig-Utschig

Clemens worked as Chief Architect for the Shared Service Centre, Global Business Services, Boehringer Ingelheim in architecture, master data, service management and innovation.

At the moment he works with holistic enterprise architecture that provides the methodological platform for the new master data management.

He previously worked as a Platform Architect at Oracle Inc. in the United States, where he helped to develop next product strategy as well as the SOA BPM Suite.

Contributions

- Event-Driven SOA
- SOA in Real Life: Mobile Solutions
- SOA and User-Interfaces
- Understanding Service Compensation
- Securing the SOA Landscape
- Enterprise Service Bus
- Canonizing a Language for Architecture: An SOA Service Category Matrix
- Industrial SOA
- SOA Blueprint: A Toolbox for Architects

Torsten Winterberg

Torsten Winterberg works for Oracle Platinum Partner OPITZ CONSULTING. As a director of the competence center for integration and business process solutions he follows his passion to build the best delivery unit for customer solutions in the area of SOA and BPM. He has long-time experience as developer, coach and architect in the area of building complex mission critical Java EE applications. He is a known speaker in the German Java and Oracle communities and has written numerous articles on SOA/BPM related topics. Torsten is part of the Oracle ACE director team (ACE=Acknowledged Community Expert) and leads the DOAG middleware community.

Contributions

- Event-Driven SOA
- SOA in Real Life: Mobile Solutions
- SOA and User-Interfaces
- Understanding Service Compensation
- Securing the SOA Landscape
- Enterprise Service Bus
- Canonizing a Language for Architecture: An SOA Service Category Matrix
- Industrial SOA
- SOA Blueprint: A Toolbox for Architects