EVENT-DRIVEN ARCHITECTURE AND SOA - ALLIES OR ENEMIES?

Bogdan Pilawski
Bank Zachodni WBK, Strzegomska 8-10, 53-611 Wroclaw, Poland
e-mail: bogdan.pilawski@bzwbk.pl

ABSTRACT:

The specialised software used to control automated plants and systems (e.g. Fly-By-Wire and aircraft flight control systems) was based on handling events from its outset. Nowadays there are more and more business conditions (business events) requiring immediate attention to cope with (e.g. on-line fraud). That brings the need to include event-handling concepts and solutions into business software applications. Software suppliers are taking different stances on that, and so are the directions and advise from research companies.

This paper undertakes to review the matter in short, concentrating on whether concepts of Service Orientated Architecture and Event Driven Architecture are separate things, or if they can be brought and act together.

KEYWORDS:

Service Orientated Architecture, Enterprise Service Bus, event, Event Driven Architecture, event handling, event processing, business process, business service

INTRODUCTION

The IT systems have been for many years designed to perform typical tasks which, for the purpose of this paper, can be divided into three major groups:

(i) Systems performing mass calculations, where the amount of input data is limited and so are the results returned. The processing of the said input data is usually long-lasting and processor intensive;

(ii) Systems processing mass data, where the amount of input data is high, usage of processing power is limited, and the volume of output data, whether its database updates or reports of various shape, is also huge. This activity can be performed on-line or in a batch manner.

(iii) Systems controlling the operation of industrial systems to ensure these behave up to the expectation, and to prevent any malfunctions, some of which, if only allowed to take place, might be of disastrous consequences in terms of loss of live, negative social impact or financial loss.

The systems of category (iii) above perform most (if not all) of their actions on-line, immediately accepting signals from the controlled plant, and processing them and issuing

commands to keep it operating up to the specification. Most of those plant-generated feedback signals indicate that something has happened, what requires attention of humans or of automated device or process. The very reason for such an indicating signal is referred to as "an event". When issued, the event must be dealt with immediately, otherwise the controlled plant may stop working properly, cease to work at all, or even go totally astray. It is well beyond comprehension to assume that – for example - a signal on a to high steam pressure in a power generating plant will be stored within a system until the daily report generation time comes, and this signal will be conveyed to a person responsible along with his/her daily report next day, and on that basis he/she will decide what action to take.[1]

The pace with which the nowadays business is run, more and more often requires for various decisions and resulting actions to be taken immediately, or almost immediately, just after business–related event has taken place. The faster the appropriate action, the better are its business results, whether the action itself is preventive or supportive in nature. The business requirement of this kind calls for an IT solution capable of conveying events to their recipients (these could be many) without any delay. The solution sought however must also ensure that the already existing IT functionality will not be affected by it.

To the extent this concept has been already present in numerous business applications, however the way it has been adopted[2] is to the contrary with latest concepts, thus preventing the implementation of methodologies like SOA, or its underlying idea of Enterprise Service Bus (ESB).

EVENT-DRIVEN ARCHITECTURE

The term "*Event-Driven Architecture*" (EDA) has been coined and introduced by Gartner in 2003, to describe a new design paradigm based on transmission of events between decoupled software components and services [Maréchaux_2006].

As defined in [Michelson_2006] paper "*an event is a notable thing that happens inside or outside your business. An event (business or system) may signify a problem or impending problem, an opportunity, a threshold, or a deviation*". What is even more important – this paper comes with the observation, that commonly the term "*event*" is often used interchangeably to refer to both the specification of the event, and also to each individual occurrence of the event.

The [Michelson_2006] paper distinguishes three main styles of event processing: simple, stream and complex.[3]

In simple event processing, when the event happens it initiates some downstream action or series of actions. It is used to drive the real-time flows to eliminate lag time and thus – to reduce cost. Stream event processing is used to propagate (stream) events to all interested

[1] However the system in question cannot over-react, calling for attention anytime the steam pressure goes up; to illustrate that, the [Chandy_2007d] paper comes with an nature example of zebra, which will die if it doesn't run away from lion (reaction to the event of spotting the lion), but it will also die from waste of energy, if it continuously keeps running away from any non-threat condition

[2] usually the event-driven element has been somehow appended to otherwise ready and working business application

[3] another (succesful) attempt to come up with deffinitions from the the area of events was made in the [Chandy_2007a] paper

parties, which subscribed to this notification, usually to support in-time decision making. Complex event processing (CEP) attempts to evaluate a confluence of events, the correlation of which may be casual, temporal or spatial. This is used to detect and to respond to business anomalies, threats, and opportunities.

Within event-driven architecture an event becomes disseminated to all parties interested (whether human or automated), and they in turn evaluate that event and decide if any action is necessary. The creator of the event has no knowledge of the event's subsequent processing, or of the parties interested in it [Michelson_2006].

Another classification has it that events may be simple, resulting from a single triggering condition becoming satisfied, or they may be complex in nature, where the final event issued results from a combination of multiple simple events. The occurrence of the latter may take place in parallel, or it may be the outcome of a cascading process.

SERVICE ORIENTATED ARCHITECTURE (SOA)

SOA is currently the culmination of a software development, it embraces procedure composition, OOP, client/server and their off-springs, from programming perspective - all based on procedure or function call. These calls are request/response in nature, and work in synch by bringing a kind of response to each request issued [Chandy_2007d]. All those components keep waiting doing nothing, until they receive a request for service.

To the contrary, the event-driven components continuously listen and sense, ready to respond to the signal or the condition, occurrence of which they are capable to discover.

The basic difference however between running services and handling events is, that a service always constitutes a request-response mechanism, i.e. the party initiating the service requests another party – a service provider, to fulfil the requirements of the request, while events are always based on unidirectional flow, just carrying the message of certain meaning, destined for un-named recipients [see: Maréchaux_2006].

The features of those two – SOA and EDA - are compared in table 1.

Table 1 SOA and EDA features

SOA	EDA
Service	Event stream
Binding	Subscription
Contract	Specification of event stream generation
Service provider	Event stream publisher
Service consumer	Event stream subscriber
Service broker	Event stream broker

Source: based on [Chandy_2007b]

Event-driven business processes and event-driven application systems help enterprises to react quickly and precisely to rapidly changing conditions [Schulte_2003b]. So far events have been widely used in system software, such as operating systems and network management systems. When used in business applications, they have been mostly limited to the so called "simple events" [Schulte_2003b].

The paper [Schulte_2003b] also has it, that there are five forces which are driving the adoption of the concept of events in business applications. These are:

1. The demand of business strategies,
2. The broad-scale migration to service-oriented architecture,
3. Vendors offering enabling tools,
4. Emerging standards,[4]
5. Hardware and network improvements.

Various papers presenting characteristics of event-driven architectures agree in principle, but they have different approach to the matter of publish/subscribe messaging. Most authors place it under EDA umbrella, while others (especially [Chandy_2007c]) perceive EDA as much more powerful, than the traditional publish-subscribe architecture, because of the flexibility and dynamic nature of contracts (specifications) between subscriber (client) and publisher (server).

To dispel any doubts one also has to stress, there is no need to run Web Services to cope with the events. Event-driven and SOA applications can be implemented with Web Services, but neither requires web services to be present. SOA message exchange patterns (MEP) already enable SOA request/reply or one-way event delivery. While in place however, ESB can extend simple, point-to-point processing of events with its services capable of providing[5]:

(i) Transport services to ensure delivery of message among the processes interconnected via ESB,
(ii) Event services to detect, trigger and disseminate events,
(iii) Mediation services to ensure matching of protocols and to offer message content transformation if required.

ALLIES OR ENEMIES?

While ESB allows for the implementation of both the SOA and the EDA concepts, EDA is sometimes also referred to as "*event-driven SOA*" [Maréchaux_2006]. What's significant – both Gartner papers quoted here - [Schulte_2003a] and [Schulte_2003b] seem to insist, EDA, despite being part of widely viewed SOA, constitutes itself a separate category of system architecture. It supplements SOA in a way, rather than being inherent part of it.

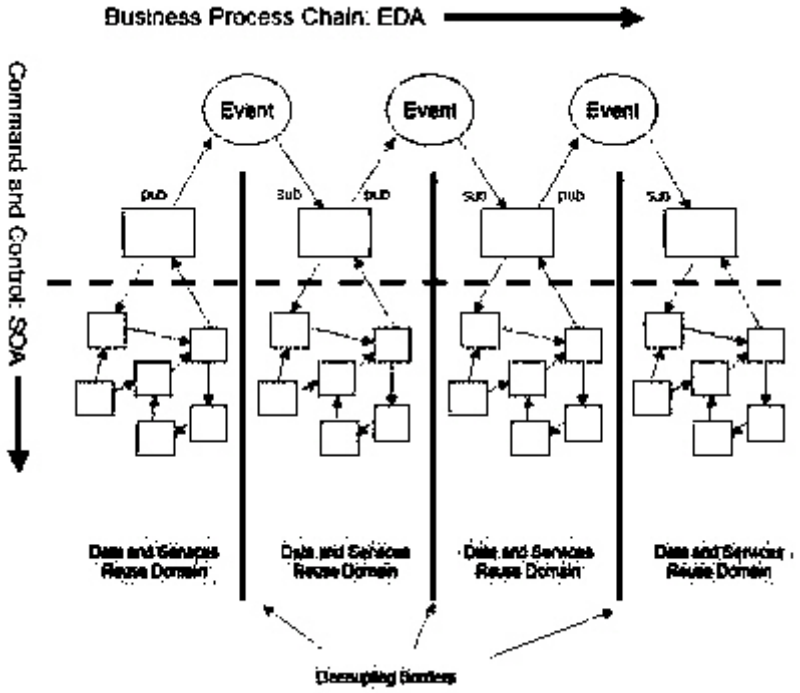
According to Gartner, the concept of events is the key factor to enable revolutionary improvements in business processes. In 2003 Gartner insisted these strategies are already promoted under labels such as "*zero-latency enterprise*", "*the real-time enterprise*" "*the event-driven enterprise*" and "*On Demand Computing*"[6] [Schulte_2003b]. Five years which passed since then show, not much of general progress in this field has been achieved, but that does not mean, the businesses and other users are not embracing event-driven architectures at

[4] the list of existing (as per March, 2007) EDA/CEP standards is presented in [SOA_2007]; the Gartner 2003 paper [Schulte_2003b] however, called these standards "incomplete".
[5] Paper [Maréchaux_2006] states there is no official specification the ESB implementation, but to be able to facilitate the integration of large-scale heterogeneous applications it must at least provide transport, event and mediation services
[6] IBM's advertising slogan

all. In financial sector they're forced into it by facts of life, like growing on-line fraud, and also – some regulatory requirements [BEA_2007].

Position similar to Gartner, separating to the extent EDA from SOA is taken also by Patricia Seybold Group [see: Michelson_2006], and is also presented in [Hoof_2006] paper. The latter however sees EDA as representing a business process chain, while SOA keeps command and control of what's going on technically. This view is presented in picture 1. The processes there are clearly separated from each other, and events provide the only link between them, carrying the process completion signal which becomes the triggering or invoking factor for the process to follow.

Picture 1. EDA versus SOA



Source: [Hoof_2006]

Entirely opposite stance has been taken by Forrester Research. From the beginning of considering EDA they insisted SOA and EDA must be seen as a one whole, and not separated in any way. They defend their position with the following: *"...vendors and pundits are discussing event-driven architecture (EDA) as a new priority and a "next big thing" requiring a separate architectural focus. To Forrester, this is a potentially costly distraction. Event-driven concepts are good and necessary, but setting up an "EDA versus SOA" distinction pulls attention away from the fundamental architectural goal of building a unified application infrastructure and a coherent set of application patterns delivered on that infrastructure*." [Heffner_2004].

In Forrester's view event handling capabilities should be treated as part of a full-featured SOA, and these should include other capabilities, such as policy-based processing, multichannel coordination, filtered message delivery, real-time business management, to name just the few most important. These all are contained within the concept of *Digital Business Architecture*, Forrester's long advocated idea of how to combine all the latest in IT architecture under one umbrella [see also: Heffner_2006]. This concept in relation to event

handling is illustrated on picture 2. It also shows the components embraced within this idea, and how do they relate to each other.

Picture 2. Forrester's functions, flows, and infrastructure for digital business event processing

Source: [Heffner_2006]

Forrester's approach seems to be more holistic in nature, when compared with Gartner's view. To the extent it results from what can be called "Forrester's event philosophy", which differs significantly from approach taken by Gartner and others. This is explained in more detail in the [Brett_2008] paper. The differences begin with the very definition of event which, as presented by Forrester, sees a number of categories which can be applied (simple. Complex, IT-related, non-IT etc.). Forrester insists the design and development of event handling software requires different skills than conventional design and programming.

CONCLUSIONS

Event driven architectures are nowadays present in the offer of almost any major IT market player. These architectures however are perceived differently, and there is no solution likely to help to answer the title question of this paper. While SOA and ESB are present in any architecture offered, from the perspective of events their role seems to be more instrumental in nature, rather than dealing with the real thing.

The [Neon_2003] paper presents IBM mainframe-related position, which seems to be limited to replacing "*pull*" approach of information dissemination, with "*push*", meaning the fact of occurrence of pre-defined events will be propagated to interested parties instead of making it just only available to them. Another IBM paper [Maréchaux_2006] refers to EDA perceived as "*event-driven SOA*".

Oracle's approach [see: Jellema_2007] is similar, however more mature, since it concentrates on the role of the Enterprise Service Bus which becomes the only means of liaison between

business processes generating events processed by business services, and these business services leading to another set of business processes.

SAP seems to limit events to business workflow-like tasks [see: Hilpert_2007], what makes it similar to "pure" Forrester's model presented earlier in this paper.

Interesting view on events has been explained in detail in the [Hohpe_2006] paper. Hohpe, who at the time of writing was one of the software architects with Google, takes very technical stance and calls EDA just "*programming without a call stack*".

All the above said views and concepts are somehow embedded within various kinds of middleware of wider than only EDA functionality[7]. Basically different position on that has been taken by BEA, which in 2007 has come up with a set of tools dedicated to processing and handling events [see: BEA_2007]. Their approach seems to be mature and very much related to the current business needs, resulting, among others, from recent regulatory requirements to financial institutions. Despite this close relationship with current business needs, BEA's proposal is far from convenient shortcuts, which might easily result should they had taken more opportunistic approach. The stance taken by BEA in relation to financial institutions seems to reply somehow to Forrester's prediction of 2006: "*While a dedicated event-driven architecture will not be necessary, banks must take precautions so that their platforms can take care of events within the framework of their SOA*" [Hoppermann_2006].

Also the organisations like OASIS, W3C and WS-I are joining in and taking steps towards providing more standards to help event-driven architecture applications to be able to talk to each other. The main areas addressed here are [Schulte_2003b]:

(i) Guaranteed delivery (WS-Reliability and WS-Reliable Messaging),
(ii) Notification specification,
(iii) WS-Addressing (publish-and –subscribe),
(iv) WSDL – standarisation on metadata.

All that however is far from a kind of soothsaying that in the near future EDA is to become "*next big thing*" (as coined by Forrester) and to take over, sending other methodologies and techniques for years present in IT to the history. Its entirely to the contrary: *"..the most viable, agile architectures will be comprised of a blend of architecture strategies, including service-oriented architecture, process-based architecture, federated information, enterprise integration and open source adoption. The best choices are the ones that match your business!*" [Michelson_2006].

[7] the particular role of event-based messaging middleware is discussed in more detail in [Rozsnyai_2007] paper

LITERATURA

[BEA_2007] *BEA WebLogic Event Server 2.0 First and Only Java Container for High-Performance Event-Driven Applications*, BEA, 2007

[Brett_2008] Brett, Charles, *What Are Events, And Why Do They Matter To Application Development Proffessionals?*, Forrester Research, 2008

[Chandy_2007a] Chandy, Mani K., Ramo, Simon, Schulte, Roy W, *What is Event Driven Architecture (EDA) and Why Does it Matter?*, http://complexevents.com, 12/2007

[Chandy_2007b] Chandy, Mani K., Carmona, Jonathan L., *Service-Oriented Architecture: Event Web Building Block,* http://www.developer.com, 12/2007

[Chandy_2007c] Chandy, Mani K., Carmona, Jonathan L., *Event-Driven Architecture vs. Publish-Subscribe Systems*, www.developer.com, 2007

[Chandy_2007d] Chandy, Mani K., Carmona, Jonathan L., *The Event Web: Sense and Respond to Critical Conditions*, www.developer.com, 2007

[Heffner_2004] Heffner, Randy, *The Unification Of SOA And EDA And More (How A Complete SOA Supports Event-Driven Applications)*, Forrester Research, 2004

[Heffner_2006] Heffner, Randy, *EDA, SOA 2.0, And Digital Business Architecture*, Forrester Research, 2006

[Hilpert_2007] Hilpert, Wolfgang, Volmering, Thomas, *Road Map for BPM and Event-Driven Architecture*, SAP AG, 2007

[Hohpe_2006] Hohpe, Gregor, *Programming Without a Call Stack-Event-driven Architectures*, www.eaipatterns.com, 2006

[Hoof_2006] Hoof, Jack von, *How EDA extends SOA and why it is important*, http://soa-eda.blogspot.com, 2006

[Hoppermann_2006] Hoppermann, Jost, *The Next-Generation Banking Platform*, Forrester Research, 2006

[Jellema_2007] Jellema, Lucas, *Building Event-Driven Architecture with an Enterprise Service Bus*, Oracle Corp., 2007

[Maréchaux_2006] Maréchau, Jean-Louis, *Combining Service-Oriented Architecture and Event-Driven Architecture using an Enterprise Service Bus*, IBM, 2006

[Michelson_2006] Michelson, Brenda M., *Event-Driven Architecture Overwiev*, Patricia Seybold Group, Boston, 2006

[Neon_2003] *A strategy for Implementing Event-Driven Architecture on IBM z/OS Mainframes,* Neon Sytems Inc, 2003

[Rozsnyai_2007] Rozsnyai, Szabolcs at al., *Concepts and Models for Typing Events for Event-Based Systems*, ACM paper 978-1-59593-665-3, 2007

[Schulte_2003a] Schulte, Roy W., Natis, Yefim V., *Event-Driven Architecture Complements SOA*, Gartner Research, 2003

[Schulte_2003b] Schulte, Roy W., *The Growing Role of Events in Enterprise Aplications, Gartner Research,* 2003

[SOA_2007] *Existing EDA/CEP Standards v.2.1*, http://soa.omg.org, March 2007