SOA & Event Driven Architecture (EDA)

Eine perfekte Symbiose

VON ROGER ZACHARIAS

SOA-Konzepte und -Infrastrukturen bilden mittlerweile die Basis für eine optimale Automatisierung von Geschäftsprozessen. Der Bereich des Geschäftsprozess-Monitorings, d.h. das Messen der Geschäftsprozesse als Basis für die Optimierung, geschieht jedoch oftmals noch ohne IT-Unterstützung, sodass erste Aussagen oft erst am Geschäftsjahresende oder im Negativfall (z.B. bei Beschwerde durch Kunden oder einem erschöpften Artikelbestand) vorliegen. Um auch diesen Bereich zu automatisieren, sind die Event Driven Architecture (EDA) und deren Kombination mit SOA die Grundlage.

Da das Thema EDA inklusive seiner Teil- und Bezugsthemen wie Event Stream Processing (ESP), Complex Event Processing (CEP), Business Activity Monitoring (BAM) und Message oriented Middleware (MOM) ebenso umfangreich wie das Thema SOA ist, sollen zunächst Ziel und Big Picture dargestellt werden, bevor auf Architekturkonzepte, die Beziehung zu SOA, Standards und notwendige Infrastrukturen eingegangen wird.

Abbildung 1 stellt den Regelkreis dar, dessen Durchführung als Business Process Management (BPM) bekannt ist (siehe Kasten). Der Regelkreis selbst ist prinzipiell nichts Neues (er existiert, seitdem es Unternehmen gibt) und hat zunächst nichts mit IT zu tun, da es hier lediglich um Planung, Steuerung, Kontrolle und Optimierung der Geschäftsprozesse eines Unternehmens geht. Es gibt zwei Gründe, warum dieser Regelkreis heute eine solch explizite Betrachtung und einen IT-Bezug erfährt: Der erste Grund ist der Wunsch, auch diesen Regelkreis selbst, also das Management der Geschäftsprozesse, immer stärker zu automatisieren (die Automatisierung begann ja bekanntlich mit dem Produkterstellungsprozess). Der zweite Grund ist die erzwungene permanente Verkürzung einer Zykluszeit dieses Regelkreises, da eine Differenzierung der Unternehmen heute immer weniger über

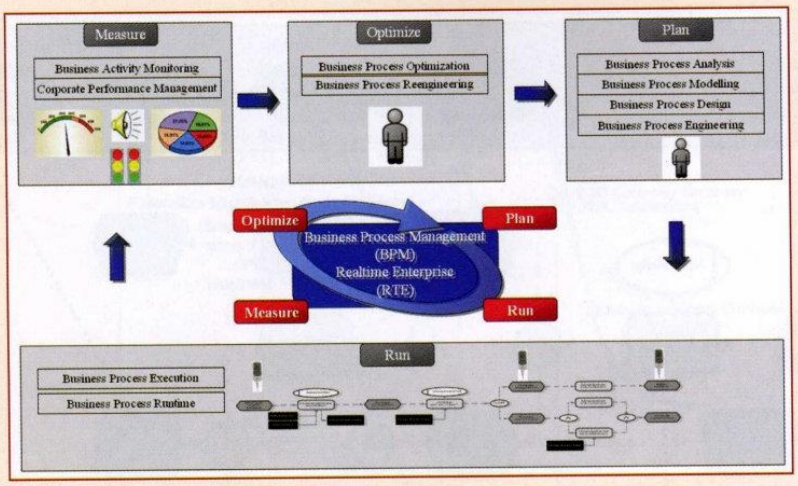


Abb. 1: Business Process Management (BPM) als Big Picture

deren Produkte nebst Qualität und Preis, sondern immer mehr über innovative und vor allem schnelle Geschäftsprozesse erfolgt.

Der Geschäftsprozess als Basis für SOA und EDA

Um uns dem Thema Event Driven Architecture (EDA) zu nähern, beginnen wir mit der Basis der meisten IT-Bemühungen – dem Geschäftsprozess. Wie in Abbildung 2 dargestellt, besteht dieser aus einer (sequentiellen/parallelen) Folge von betriebswirtschaftlichen Aktivitäten (Funktionen) und Geschäftsereignissen. Hierbei wird jede Aktivität durch ein Geschäftsereignis angestoßen und jede durchgeführte Aktivität resultiert in einem Geschäftsereignis.

Jede Aktivität, welche das Zeit- und Kosten-konsumierende Element einer Prozesskette darstellt, wird mit ihren Eingabe-/Ausgabe-Daten auf eine Service-Operation eines SOA-Services abgebildet. Dieser führt die fachliche Funktion durch und transferiert dabei Eingabedaten in Ausgabedaten. Aber was passiert mit den Geschäftsereignissen, welche neben den Funktionen ein zentrales Prinzip der Geschäftsprozessmodellierung sind? Diese finden innerhalb einer Service orientierten Archtitektur nur implizit Berücksichtigung, sind aber das zentrale Element einer EDA. Denn in einer ereignisgesteuerten Architektur geht es um das Erkennen, Auslösen, Empfangen, Verarbeiten von und die Reaktion auf Ereignisse unterschiedlichster Art. Das heißt, EDA rückt die Geschäftsereignisse nun ins Zentrum der Architektur. In einer EDA haben die Ereignisse (wie die Services in einer SOA) fachlichen und damit geschäftsrelevanten Charakter. Beispielsweise stößt das Ereignis „Auftragseingang" die order-to-cash Prozesskette an und die resultierenden Ereignisse „Ware im Lager eingetroffen", „Ware versendet" und „Rechnung versendet" liefern wichtige Informationen über den aktuellen Prozesszustand.

Einsatzgebiete von EDA

Die Haupteinsatzgebiete der EDA in kommerziellen Anwendungssystemen sind die Verwendung von Ereignissen als fundamentaler Bestandteil der Prozesskette und damit der Geschäftslogik sowie die

Realtime Enterprise (RTE)

Der zu erreichende Zielzustand eines Unternehmens ist das von Gartner propagierte Realtime Enterprise (RTE), welches sich durch einen maximalen Automatisierungsgrad, minimale Prozesslaufzeiten und möglichst wenig menschliche Interaktion innerhalb der Geschäftsprozesse auszeichnet. Dies kann nur durch einen permanenten, iterativen Regelkreis der Aktivitäten „Plan", „Run", „Measure" und „Optimize" bzgl. der wertschöpfenden und administrativen Geschäftsprozesse eines Unternehmens erreicht werden, das heißt mittels aktivem Management der Geschäftsprozesse selbst (Business Process Management). In den Bereichen Planung und Optimierung, welche bislang kaum automatisierbar sind, sind die Konzepte der Geschäftsprozessanalyse, -modellierung, -reengineering usw. neben entsprechenden Werkzeugen und Notationen (z.B. ARIS) wohlbekannt.

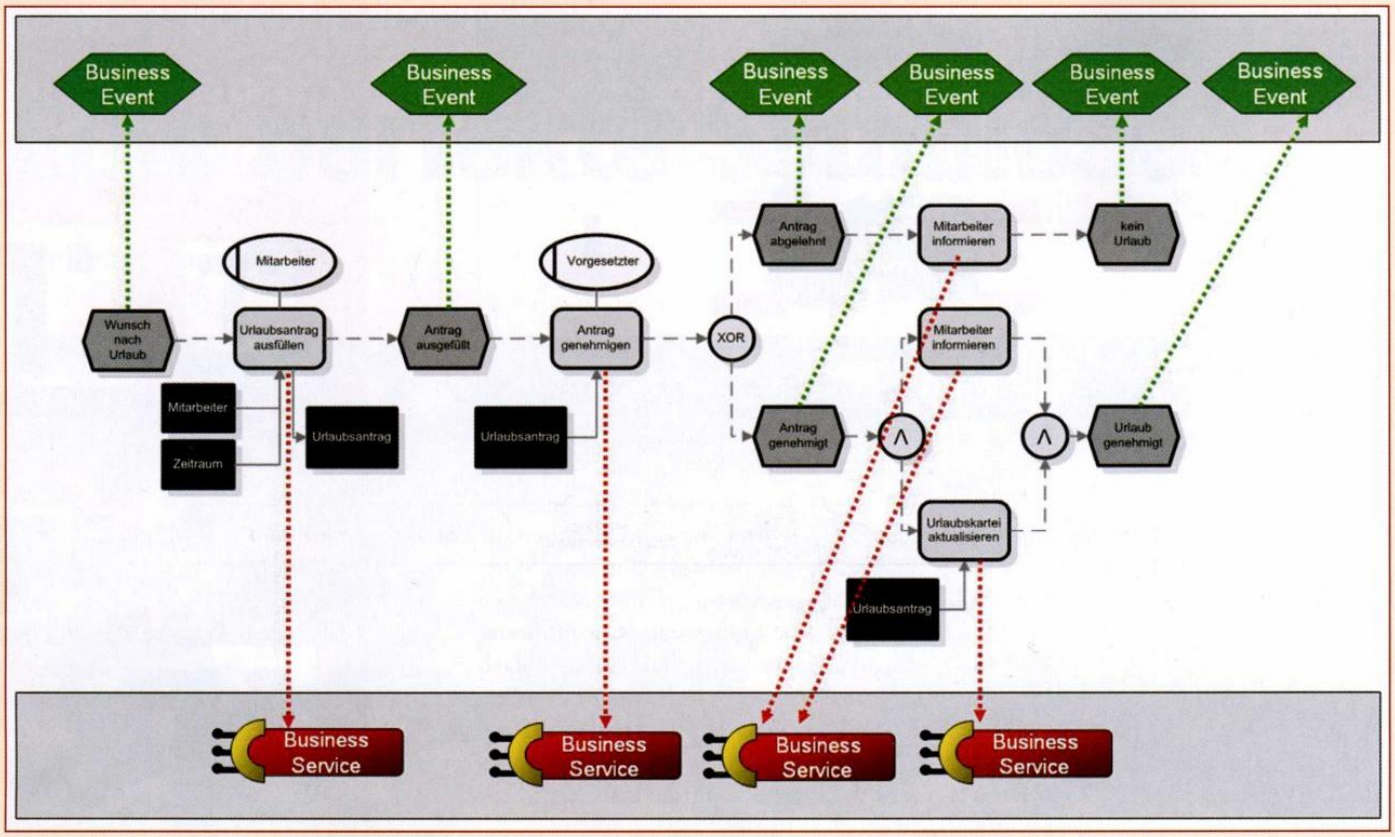


Abb. 2: Der Geschäftsprozess als Basis für SOA und EDA

Verwendung der Ereignisse, die innerhalb der Prozesskette erzeugt werden, für das Echtzeit-Monitoring des Prozesses, auch als Business Activity Monitoring (BAM) bezeichnet.

Die Ereignisverarbeitung selbst ist nichts Neues und seit vielen Jahren insbesondere in Betriebssystemen, Embedded-Systemen, in Publish-Subscribe-Szenarien und im System Management erfolgreich im Einsatz. Bisher wurde aber (bis auf den Einsatz in Spezialbereichen wie dem Börsenhandel) selten das Potenzial für Business-Applikationen erkannt. Dies zeigt sich insbesondere in den fehlenden Standards und Produkten in diesem Umfeld trotz den seit Jahren vorhandenen Infrastrukturen und Infrastrukturkonzepten.

Das Neue an der Ereignisverarbeitung sind die stark fachliche Ausrichtung, die Kombination mit SOA-Konzepten, die beginnende Standardisierung, erkannte Anwendungsszenarien und vor allem das steigende Bewusstsein und Interesse für EDA. Deutlich wird dies durch die zunehmende Anzahl von Initiativen, Arbeitsgruppen, Publikationen und vor allem die Aufnahme des Themas in den Gartner-Hypecycle (siehe Abb. 3).

Architekturbestandteile von EDA

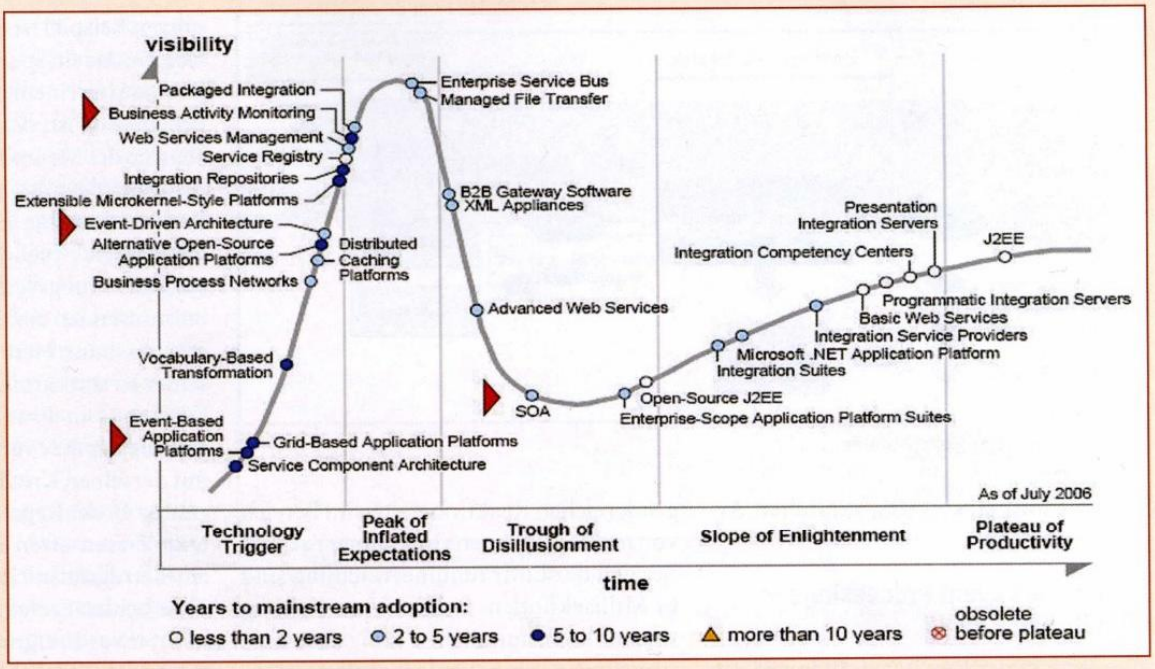
Das zentrale Element der EDA ist das Ereignis (Event), welches eine aufgetretene Situation innerhalb eines technischen oder organisatorischen Systems beschreibt und damit der Informationsträger bzw. die Signatur und Dokumentation dieser Situation ist. Hierbei kann eine grobe Klassifikation in technische und fachliche Ereignisse (Business Events) vorgenommen werden.

Einige Beispiele für technische Ereignisse sind: Unterschreitung eines definierten Schwellenwerts der CPU-Nutzung, eine gestiegene Speicherauslastung auf 95 Prozent, Ausfall eines Servers, Überschreitung des SLA für Antwortzeit, Airbag ausgelöst oder Papierleerlauf eines Kontoauszugsdruckers. Beispiele für fachliche Ereignisse sind Unterschreitung eines Lagerbestandes, Eintreffen einer Bestellung, Stornierung eines Auftrages, Insolvenz eines Lieferanten, Ware verlässt Lager, Aktienkurs unterschreitet Schwellenwert oder Kundenkonto angelegt.

EDA-Systeme sind also Systeme, in denen derartige Situationen erkannt, entsprechende Ereignisse erzeugt, versendet und von Interessenten entgegengenommen, ausgewertet und daraufhin entsprechende Aktionen angestoßen werden. Folgende Architekturkomponenten (siehe Abb. 4) sind daher Bestandteil einer EDA:

- Event Sensor: Diese Komponente ist für das Erkennen der aufgetretenen Situation zuständig und kann je nach Art des Systems und der zu erkennenden Situation selbst sehr komplex sein.
- Event Producer: Diese Komponente ist im ersten Schritt für das Generieren des Ereignisses gemäß der aufgetretenen Situation und im zweiten für das Versenden desselben auf den Event Channel zuständig. Hierbei können bereits Filterungen durchgeführt werden.
- Event Channel: Diese Komponente stellt die Infrastruktur für den Transport der Ereignisse dar, wobei sowohl mehrere Event Producer als auch Event Consumer angeschlossen sein können und ein Ereignis auch mehrere Empfänger haben kann

Abb. 3: Gartner-Hypecycle 2006 (Quelle: [6])



(publish/subscribe-Prinzip). Die Gesamtheit aller Event Channels in einem System wird als Event Bus bezeichnet.

- Event Consumer: Diese Komponente ist für das Empfangen der auf den Event Channel publizierten Ereignisse zuständig, wobei in der Regel auch hier eine entsprechende Filterung erfolgt.
- Event Processor: Diese Komponente ist für das Verarbeiten der empfangenen Ereignisse zuständig. Die Form der Verarbeitung kann hierbei sehr unterschiedlich ausfallen (Ignorieren, Vermerken, weitere Analysen, Informieren menschlicher Empfänger oder Systeme, Erzeugen neuer Ereignisse, Ausführung von Aktionen oder Anstoß ganzer Prozesse), wodurch die Komponente stark unterschiedliche Ausprägungen haben kann.

Wie man sieht, ist dieses Grundprinzip auf unterschiedlichste Systeme anwendbar: Dies reicht z.B. von der menschlichen Kommunikation (Event Sensor ist das Auge der Person A, Event Producer Gehirn und Stimme von A, Event Channel die Luft zum Übertragen der Stimme, Event Consumer das Ohr der Person B und Event Processor das Gehirn von B) bis zu komplexen Börsensystemen, die basierend auf gemeldeten Aktienschwankungen ohne menschliche Interaktion Aktien kaufen oder verkaufen.

Oft können EDA-Systeme damit die Realität besser abbilden als Systeme, welche lediglich Standard-Kontrollflüsse ohne Ereignisse verwenden, denn die reale Welt ist meist ereignisgetrieben: Ein Auftragseingang löst den zugehörigen order-to-cash Prozess aus, die Threshold-Information aus dem Lager löst eine Nachbestellung aus, die Ehefrau ruft zum Abendessen und unterbricht den gerade ablaufenden Kontrollfluss des Artikelschreibens, der Wecker

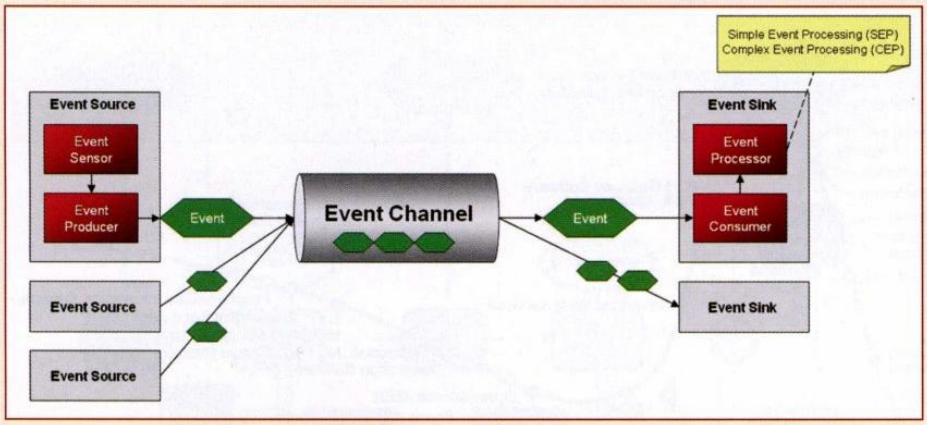


Abb. 4: EDA-Architekturkomponenten

klingelt und startet den täglichen Arbeitsprozess etc.

Formen des Event Processings – EP, ESP, SEP, CEP

Wie auch in der Realität stellt innerhalb eines EDA-Systems die Verarbeitung der eintreffenden Ereignisse die größte Herausforderung dar. Je nach organisatorischem bzw. technischem System treten die Ereignisse in unterschiedlichster Form und vor allem Häufigkeit auf, was sich stark auf die zu verarbeitende Einheit (Event Processor) auswirkt. Im einfachsten Fall gibt es nur wenige unterschiedliche Ereignistypen, es treffen nie parallel zwei Ereignisse ein und zwischen den Ereignissen liegen Minuten (z.B. Ereignisse eines Temperatursensors).

Betrachtet man aber als Beispiel einen zentralen Event Bus eines weltweiten Klimaforschungsinstitutes, an den tausende Temperatursensoren und viele andere Sensoren (für Windgeschwindigkeiten Luftfeuchtigkeit, Meeresströmungen usw.) angeschlossen sind, entsteht eine extreme Dynamik aus vielen verschiedenen Ereignistypen und hunderten parallel eintreffenden Ereignissen pro Sekunde. In diesem Fall handelt es sich nicht mehr um Einzelereignisse, sondern um komplexe kontinuierliche Datenströme. Dieselben Rahmenbedingungen entstehen auch bei zentralen Logistiksystemen, Kernbankensystemen, militärischen Überwachungssystemen oder Flugverkehrskontrollsystemen. Hierbei bestehen je nach System weiterhin unterschiedliche Anforderungen an die Latenz zwischen Auftreten des Ereignisses und Reaktion auf dieses. Im Bereich der Logistik reichen Reaktionszeiten im Bereich von mehren Minuten vollkommen aus, im Bereich der Luftraumüberwachung sind es Millisekunden. Bei der Verarbeitung solcher kontinuierlichen Ereignisströme spricht man von Event Stream Processing (ESP) im Gegensatz zum einfachen Event Processing (EP).

Bei beiden Formen existieren nun weiterhin jeweils zwei grundlegende Möglichkeiten der Ereignisverarbeitung. Die einfache Form, das Simple Event Processing (SEP), betrachtet jedes eintreffende Ereignis lediglich in Abhängigkeit von dessen Typ, Inhalten und Auftrittszeitpunkt, aber unabhängig von allen anderen Ereignissen. Basierend auf einer definierten Regelbasis kann nun eine zugehörige (Re)Aktion durchgeführt werden, zum Beispiel eine Temperaturanpassung der Heizung aufgrund der am Sensor festgestellten Temperaturänderung oder die Information des Kunden nach Versand der bestellten Ware.

Oft ist aber die Sequenz und der Zusammenhang mehrerer Ereignisse viel wichtiger bzw. interessanter als das einzelne Ereignis: Aus den drei einzelnen Ereignissen „Läuten von Kirchenglocken", „Geschmücktes Auto" und „Mann in Smoking", welche in einem kurzen Zeitrahmen auftreten, kann das komplexe Ereignis „Hochzeit" abgeleitet werden. Die Ereignisse könnten aber auch unabhängig voneinander stehen (vgl. [4]). Werden nun statt der Einzelbetrachtung die Ereignisse miteinander in Beziehung gesetzt (korreliert) und hierbei komplexe Muster entdeckt, spricht man von Complex Event Processing (CEP). Der Event Processor im obigen Beispiel ist dabei das menschliche Gehirn, das ein Spezialist für die Mustererkennung in einem gewaltigen Strom von Ereignissen ist, der durch die fünf Event Sensors des Menschen erzeugt wird.

Das folgende (in Abb. 5 dargestellte) Beispiel soll den Einsatz im Unternehmensbereich (vgl. [4]) verdeutlichen: In einem Zahlungsverkehrsystem treten - neben tausenden anderen - folgende Einzelereignisse auf: Herr Schneider bezahlt mit seiner Kreditkarte einen Artikel in einem Kaufhaus in Frankfurt/Main. Wenige Minuten später versucht Herr Schneider mit derselben Kreditkarte in Tokio zu bezahlen. In der Regel erkennen heutige Systeme diesen offensichtlichen Kreditkarten-Betrugsfall nicht. Mittels CEP können diese beiden Ereignisse korreliert und ein komplexes Ereignis „Betrugsverdacht" generiert werden. In einer durchgängig ausgebauten EDA würde dieses wiederum nicht nur protokolliert, sondern zusätzlich eine entsprechende Aktion, nämlich das Sperren der Kreditkarte, anstoßen.

Complex Event Processing

Die Hauptschwierigkeit besteht in der Verarbeitung der gewaltigen Menge an unterschiedlichen Ereignissen, weshalb der Ereignisstrom auch als Ereigniswolke oder Ereignistornado bezeichnet wird. Theoretisch betrachtet stellt die Gesamtheit aller im Unternehmen auftretenden Ereignisse innerhalb der Ereigniswolke zu einem Zeitpunkt den aktuellen Zustand aller Unternehmensprozesse dar. In dieser Ereigniswolke stecken – wie auch in den Unternehmensdatenbeständen – interessante Informationen, welche meist noch nicht genutzt werden, sodass man hier parallel zur Disziplin des Data Mining eine Disziplin des Event Mining etablieren könnte.

Um Anwendungsfälle realisieren zu können, ist es in einer EDA trotz des Ereignistornados notwendig, die Ereignisse in Echtzeit (nicht in nachfolgenden nächtlichen oder monatlichen Batches) zu verarbeiten, d.h. Muster zu erkennen und entsprechende sofortige Reaktionen und Prozesse anzustoßen. Hierfür sind die traditionellen RDBMS-lastigen Anwendungen, welche die Ereignisse zunächst in einer Datenbank zur späteren Analyse ablegen, natürlich nicht geeignet, da allein die

Anwendung der ACID-Kriterien für jedes einzelne Ereignis zu hohe Latenz erzeugen, sodass ein derartiges System nicht skaliert. Daher wird im EDA/CEP-Umfeld das traditionelle Verarbeitungsmodell invertiert: Statt die Daten (hier Ereignisse) in einer Datenbank abzulegen und später über Queries abzufragen, werden hierbei die Queries (Muster und zugehörige Reaktionsregeln) in der Datenbank abgelegt und die Daten (Ereignisse) auf diese angewendet. Die permanente Evaluierung der Ereignisströme gegen die hinterlegten Muster und den Anstoß der Aktionen übernimmt eine entsprechende ESP/CEP-Engine.

Für die Beschreibung der im Ereignisstrom aufzufindenden Muster aus Einzelereignissen reichen typische Programmier- oder Abfragesprachen nicht aus, da dabei insbesondere der temporale Aspekt nicht berücksichtigt wird. Obwohl diese Sprachen noch ein beliebter Forschungsgegenstand sind, existieren bereits einige spezifische Event Processing Languages (EPL) und Event Query Languages (EQL), die meist deklarativen Charakter haben.

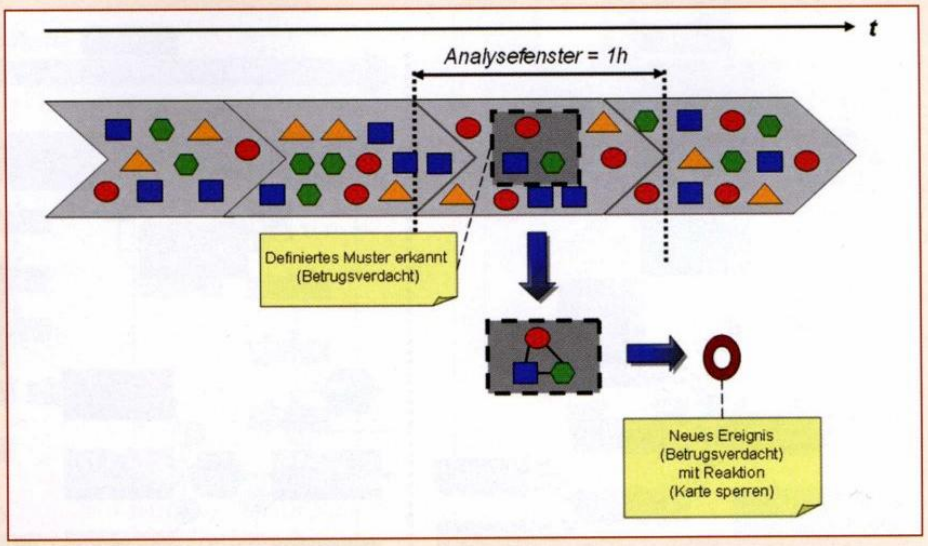


Abb. 5: Complex Event Processing (CEP)

Diese Sprachen ermöglichen trotz ihres einfachen Aufbaus fast beliebige Beschreibung von temporalen, spatialen und kausalen Zusammenhängen (Mustern, Korrelationen) von Ereignissen. Zu den temporalen Zusammenhängen gehören zum Beispiel Ausdrucksmittel bzgl. der Reihenfolge von Ereignissen, ihrem zeitlichen Abstand, dem Ausbleiben von Ereignissen in einem definiertem Zeitfenster oder ihrem Bezug zur aktuellen Zeit. Die spatialen Zusammenhänge ermöglichen

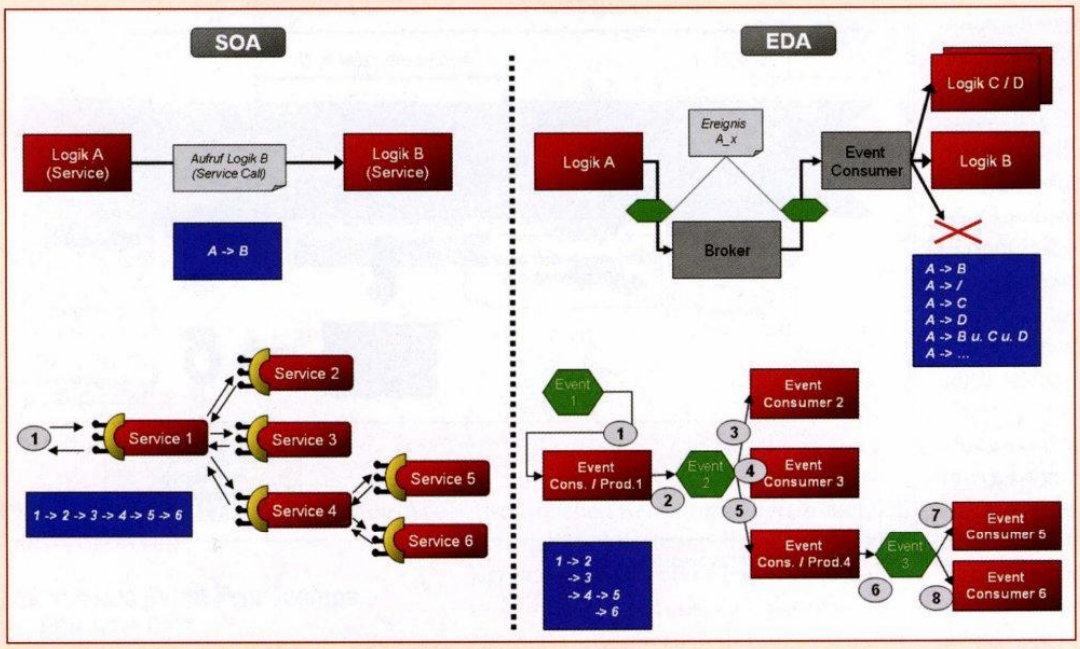


Abb. 6: Der Kontrollfluss in SOA und EDA

die Interpretation auf Basis des Auftrittsortes bzw. der Auftrittsregion und die kausalen Zusammenhänge die Interpretation beliebiger weiterer Kontextinformationen der Ereignisse, welche größtenteils aus den Inhalten der Ereignisse stammen: Ereignistyp, Auslöser, verursachendes und zugehöriges verursachtes Ereignis, Ereignishierarchien, Duplikaterkennung, Filter, Thresholds usw. Das komplexe Ereignis „Hochzeit" könnte mit diesen Sprachmitteln zum Beispiel folgendermaßen beschrieben werden: ON (*person_appeared* WITH person.gender EQUALS „men" AND person.clothes EQUALS „smoking") AND (*car_appeared* WITH car.appearence EQULAS „remarkable") FOLLOWED BY *church_bell* WITHIN 2 hours ACTION (ALERT *wedding*).

Vor- und Nachteile einer EDA

Der EDA-Ansatz bringt eine Reihe von Vorteilen gegenüber typischen sequentiellen Systemen mit sich: Der Hauptvorteil besteht in einer besseren Abbildbarkeit der realen, meist ereignisgetriebenen Welt mit sehr hoher Dynamik, sodass die Durchlaufzeiten von automatisierten Prozessen drastisch reduziert werden können und in Echtzeit auf Ereignisse reagiert werden kann.

Weiterhin erfolgt in einer EDA eine Entkopplung der Kommunikationspartner, sodass diese unabhängig voneinander bestehen können: Der Sender eines Ereignisses muss weder wissen, wer in welcher Form das Ereignis verarbeitet, noch um wie viele Empfänger es sich handelt. Der Empfänger eines Ereignisses braucht keine Kenntnis über den oder die Sender. Lediglich die Form und Inhalte der ausgetauschten Ereignisse müssen beiden Seiten bekannt sein. Damit können Systeme, die in unterschiedlicher Form realisiert sind, verschiedene Entwicklungszyklen und Systemgrenzen besitzen und in unterschiedlichen Verantwortungsbereichen liegen, einfacher miteinander kommunizieren. Zusätzlich sind durch diese Entkopplung Punkte wie Verarbeitungsparallelität (ein Ereignis stößt parallele Aktivitäten multipler Empfänger an), die Konfiguration der Kommunikationspartner (dynamisches Hinzufügen von Sendern und Empfängern) und die Abbildung von m:n-Beziehungen von Sendern und Empfängern einfacher abbildbar.

Natürlich ist EDA nicht in jedem beliebigen Kontext anwendbar und hat auch Nachteile – insbesondere in der Entwicklung derartiger Systeme. Oder um es mit Martin Fowlers Worten zu beschreiben „the architect´s dream, the developer´s nightmare". Durch die Entkopplung der Systeme und insbesondere durch den Verlust des Call-Stacks (s.u.) können Probleme erst zur Laufzeit erkannt werden, wodurch die Systeme schwerer zu entwickeln und zu testen sind. Dadurch dass in EDA-Systemen an unterschiedlichen Stellen Ereignisse erzeugt und konsumiert werden, die wiederum zu Aktionen oder neuen Ereignissen führen, ist die Visualisierung und das Verständnis der Kontrollstrukturen deutlich komplexer und stellt daher trotz der besseren Abbildung der Realität oft höhere Anforderungen an die Skills der Architekten und Entwickler.

Vergleich von SOA und EDA

Was unterscheidet nun eine Service-orientierte von einer Ereignis-getriebenen Architektur? Unabhängig von der Architektur selbst natürlich ihre Position im Gartner-Hypecycle (siehe Abb. 3), welche sich auch auf die Anzahl der Treffer in Google abbilden lässt (Service oriented Architecture: 20 Millionen Treffer, Event driven Architecture: 200.000, also Faktor 100). Auch wenn EDA in manchen Publikationen als Nachfolger von SOA dargestellt wird, der die in SOA existierenden Herausforderungen löst, sind SOA und EDA letztendlich zwei orthogonale Ansätze, welche vollkommen unabhängig voneinander existieren können, d.h. es kann SOA ohne EDA und EDA ohne SOA geben. Optimal ist letztendlich die Kombination beider Architekturen, in der EDA als nicht-invasive Erweiterung einer SOA verwendet wird.

Der architekturelle Hauptunterschied besteht im Grad der Kopplung der Kom-

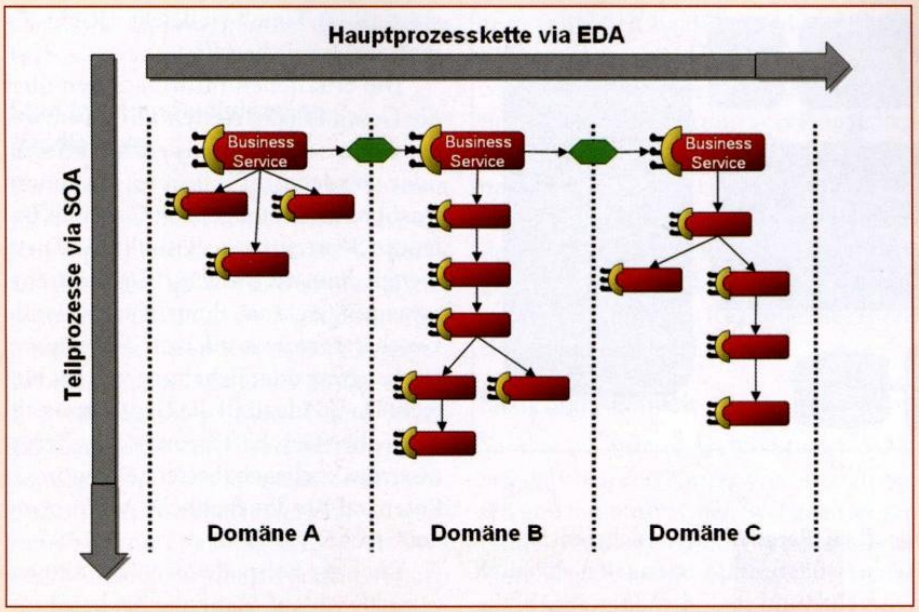


Abb. 7: EDA zur Entkopplung von SOA-Domänen

munikationspartner: Während in einer SOA eine lose Kopplung realisiert ist (der Service Consumer kennt den Service Provider, der Provider den Consumer nicht mehr), sind in einer EDA die Kommunikationspartner vollständig entkoppelt (lediglich die Ontologie der ausgetauschten Ereignisse ist beiden bekannt). Dadurch ermöglicht die EDA im Vergleich zu einer SOA mit ihrem 1:1-Verhältnis zwischen den Kommunikationspartnern zusätzlich auch 1:n- und m:n-Verhältnisse und die Konfiguration der Kommunikationspartner zur Laufzeit.

In Abbildung 6 wird der Kopplungsgrad von SOA und EDA anhand des jeweils typischen Kontrollflusses verdeutlicht. Man sieht, dass in einer SOA immer der aufrufende Service über den nächsten Logik-Schritt entscheidet (Orchestrierung/Choreografie) und dessen Abarbeitung abwartet, wodurch ein synchronisierter sequentieller Kontrollfluss entsteht. In einer EDA wird die Entscheidung über den nächsten Logik-Schritt an den Empfänger abgegeben, der Sender informiert lediglich über eine aufgetretene Situation. Weiterhin kann hierbei der Empfänger nicht nur einen definierten Folgeschritt anstoßen, sondern hat noch weitere unterschiedliche Möglichkeiten: Er kann das eintreffende Ereignis komplett ignorieren, sodass keine weitere Verarbeitung stattfindet, er kann zwischen unterschiedlichen nachfolgenden Logik-Schritten wählen oder es können mehrere Logik-Schritte parallel ablaufen. Dadurch entsteht in der Regel ein asynchroner, paralleler Kontrollfluss.

Insgesamt sieht man, dass beide Architekturen (SOA als „pull architecture" und EDA als „push architecture") sehr unterschiedlich sind und je nach Einsatzgebiet spezifische Vor- und Nachteile besitzen. Zur Realisierung des am Anfang des Artikels beschriebenen Big Pictures ist aber letztendlich eine geschickte Kombination bzw. Symbiose aus SOA- und EDA-Konzepten notwendig. In den unterschiedlichen Visionen der Hersteller hat diese Kombination Namen wie „Next Generation SOA", „Advanced SOA", „Event Driven SOA", „Realtime SOA" oder gar „SOA 2.0".

Kombination von SOA und EDA

Die Kombination von SOA und EDA lässt sich am einfachsten dadurch realisieren, dass die SOA bezüglich ihrer Grundsätze bestehen bleibt und durch EDA-Aspekte nicht-invasiv erweitert wird. Das heißt die Geschäftslogik liegt weiterhin in Form von kombinierbaren, fachlichen, Consumer-unabhängigen Services vor, aber die auslösenden und resultierenden Ereignisse der Prozesskette werden in EDA-Ereignisse abgebildet, welche dedizierte

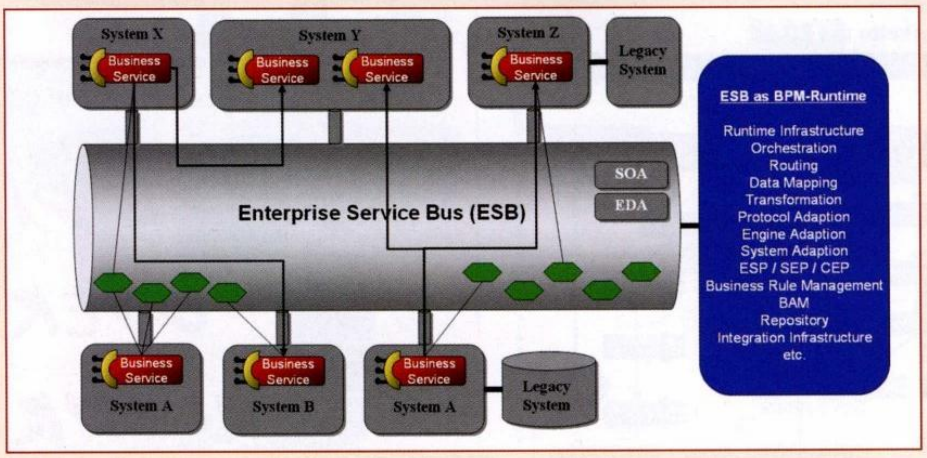


Abb. 8: Der ESB als Backbone für SOA und EDA

Situationen im Geschäftsablauf widerspiegeln.

Die Basis der EDA-Erweiterung bildet das Versenden von Geschäftsereignissen (welche aufgetretene Geschäftssituationen beschreiben) durch die Services (z.B. durch Aufruf eines spezifischen technischen Event Producer Services der das Event auf einen Bus publiziert), die damit indirekt zum Event Producer werden. Damit liegen alle Geschäftsereignisse nun auf dem Event Bus vor und können im Rahmen aller beschriebenen EDA-Konzepte wie Event Stream Processing, Complex Event Processing etc. in Echtzeit verwendet werden, was enormes Potenzial birgt und letztendlich die SOA erst komplettiert.

Beispielsweise können - unabhängig vom eigentlichen Prozess - menschliche und technische Stakeholder über die aufgetretene Situation (z.B. Ware wurde versendet) informiert und diese entsprechend protokolliert werden (Audit Trail). Innerhalb des Prozesses können die Ereignisse zur Verknüpfung der Services verwendet werden, d.h. ein von einem Service versendetes Ereignis kann einen anderen Service oder einen gesamten Prozess asynchron anstoßen. Dies bedeutet, dass ein Service nicht mehr nur durch einen anderen Service aufgerufen, sondern auch durch ein Ereignis angestoßen werden kann, also als Event Consumer fungiert.

Auch ein Anstoß mehrerer unabhängiger Services ist möglich, welche anschließend parallel und unabhängig voneinander ablaufen. Diese Konstrukte erweitern die Möglichkeiten des typischen synchronen, sequentiellen SOA-Kontrollflusses deutlich (Parallelität, Asynchronität etc.), sodass bestimmte Szenarien deutlich einfacher und realitätsnäher abgebildet werden können. Vor allem lassen sich mit dieser Kombination sehr schön die in einer SOA oft entstehenden langen Prozessketten sowie Domänen-übergreifende Prozesse entkoppeln (siehe Abb. 7): Teilprozesse innerhalb einer Domäne, welche eine sequentielle Abarbeitung kohäsiver Funktionen verlangen, werden weiterhin über normale Service-Orchestrierung abgedeckt. Zwischen den Teilprozessen oder Domänen werden Geschäftsereignisse ausgetauscht, welche den nächsten Teilprozess anstoßen und dadurch eine Entkopplung bewirken.

Neben der Nutzung der auf dem Event Bus vorliegenden Geschäftsereignisse für Informations- und Protokollierungsaktivitäten, sowie innerhalb der Prozesskette selbst, liegen sicher die interessantesten Anwendungsmöglichkeiten des so genannten Business Activity Monitoring (BAM).

Business Activity Monitoring (BAM)

Unter BAM versteht man das Monitoring der laufenden Geschäftsaktivitäten in Echtzeit (Geschäftstätigkeitsüberwachung). Echtzeit bedeutet im Bereich der Geschäftsprozesse nicht verzögerungsfrei, sondern dass die Informationen zum richtigen Zeitpunkt vorliegen, sodass im Business-Sinne rechtzeitig die notwendigen Aktionen durchgeführt werden können. Ein gutes Beispiel ist die Information über den Leerlauf eines Produktbestandes, sodass rechtzeitig eine Produktnachbestellung erfolgen kann. Also wäre statt „Real-Time" vielleicht „Right-Time" der bessere Begriff.

Die erhaltenen Informationen über die Geschäftsaktivitäten führen entweder direkt und automatisiert zu Aktionen oder werden grafisch in so genannten Dashboards, Management Cockpits, Balanced Scorecards etc. visualisiert. Diese dienen dann als Basis für Geschäftsentscheidungen, zum Eingriff in laufende Geschäftsprozesse oder zur Erkennung, Vorbeugung oder Behebung von Problemen. Da im Idealfall die Geschäftsereignisse aller Geschäftsprozesse als Ereignisstrom vorliegen, besteht ein enormes Potenzial für das fachliche Monitoring der Geschäftsprozesse.

Ein paar Beispiele hierfür sind: prozentuale Ablauf-Häufigkeiten bestimmter Prozesspfade (Teilprozesse), End-to-End-Prozesslaufzeiten, Durchlaufzeiten oder Lagerzeiten von Waren, Rüstzeiten, Prozesskosten, aktuelle Verkaufszahlen verschiedener Produkte, Reaktionszeiten auf Anfragen, Wartezeiten im Prozess usw. Um Vergleichswerte („Was sind schlechte, gute, kritische Werte? Was heißt teuer, schnell, gut?") für die Unternehmens- bzw. prozessspezifische Beurteilung, Reaktion, Ziele und Optimierungen festzuschreiben, werden so genannte Key Performance Indicators (KPIs) definiert, welche nun als Ausgangsbasis für z.B. SLA-Management verwendet werden.

BAM kann damit als Evolution bzw. Ergänzung des typischen Reportings, Business Intelligence (BI) und Data Warehousing gesehen werden: Während diese Ansätze in die Vergangenheit des Geschäfts schauen, betrachtet BAM die Gegenwart. Damit können wichtige Geschäftsentscheidungen innerhalb kürzester Zeit - und nicht erst nach dem monatlichen Abgleich der operativen Daten mit dem Data Warehouse und Versand des Monatsberichts - getroffen werden. Beispielsweise möchte man bei einer zweiwöchentlichen Kampagne jeden Abend wissen, wie diese gelaufen ist, sodass man ggf. für den nächsten Tag Waren oder Personal nachbestellen kann. Mit den klassischen Vorgehensweisen kann man lediglich am Ende des Monats herausfinden, wie erfolgreich die gesamte Kam-

pagne war und was man bei der nächsten verbessern kann.

Standards und notwendige Middleware

Da es sich bei den dargestellten Themen (SOA, EDA, ESP, CEP, BAM etc.) um Konzepte handelt, existieren für deren Umsetzung unterschiedlichste technische Plattformen. Zu dieser Middleware gehören Application Server, Rule Engines, Business Process Engines, MQ-Broker, ESP/CEP-Engines sowie BI- und BAM-Werkeuge. Obwohl diese Infrastruktur zum Teil schon seit Jahrzehnten vorhanden ist, hat sich deren integrierte Verwendung zum durchgängigen Business Process Management aus diversen Gründen noch nicht durchgesetzt:

Zum einen macht die Verwendung von JBoss AS allein die darauf ablaufenden Anwendungen nicht automatisch zu Service orientierten Architekturen, genauso wenig wie die Verwendung von MQ Series alle angeschlossenen Anwendungen zu Ereignis-gesteuerten Architekturen macht. Hier ist also eine weitere Standardisierung der Konzepte notwendig. Zum anderen sind die Middleware-Produkte heute weder innerhalb einer Hersteller-Suite alle vorhanden und integriert, noch lassen sich die Produkte unterschiedlicher Hersteller problemlos miteinander integrieren. Hier ist also eine weitere Standardisierung der Technik notwendig. Nichtsdestotrotz können und sollten heute die ersten Schritte (nicht nur in Spezialbereichen) gegangen werden. Hierzu reicht z.B. ein einfacher J2EE-Applikationsserver aus, da dieser im Standardumfang sowohl einen „SOA-Bus" (RMI/IIOP- und WebService-Infrastruktur) als auch einen „EDA-Bus" (JMS-Infrastruktur) beinhaltet.

Momentan existieren leider kaum Industriestandards in den Bereich SOA und EDA, aber Standardisierungsbemühungen wie das SOA Reference Modell, Common Event Infrastructure (CEI), Common Base Event (CBE), WS-Event Notification oder Java Business Integration (JBI) zeigen zumindest die ersten Ansätze. Technisch wird voraussichtlich der standardisierte Enterprise Service Bus (ESB) die genannten Middleware-Produkte unterschiedlicher Hersteller vereinigen und damit zum zentralen Backbone für SOA und EDA werden (siehe Abb. 8).

Mit fortschreitender Standardisierung und Kommoditisierung von Konzepten und Technik sowie dem damit verbundenem Skill-Erwerb von Entwicklern, Architekten und Business-Analysten rückt der in Abb. 1 dargestellte Regelkreis aus „Plan", „Run", „Measure" und „Optimize" und damit das Realtime Enterprise (RTE) in greifbare Nähe.

Fazit und Ausblick

Service-orientierte Architekturen (SOA) ermöglichen den Sprung von zentralisierten und monolithischen Systemen zu verteilten, modularen und lose gekoppelten Systemen, die eine Abbildung und Flexibilität von Geschäftsprozessen ermöglichen. Ereignis-gesteuerte Architekturen (EDA) ergänzen diese Aspekte um eine optimale Abbildung von Geschäftsereignissen und die Möglichkeit, Geschäftsprozesse und deren Performance in Echtzeit messen und auswerten zu können, sodass darauf basierend notwendige Entscheidungen in kürzester Zeit getroffen und Optimierungspotenzial früh erkannt werden kann.

Eine Kombination von SOA und EDA zu einer Service-orientierten und Ereignis-gesteuerten Architektur bildet damit die Grundlage für ein erfolgreiches Business Process Management und ist unabdinglich zur Realisierung des Realtime Enterprise.

Dipl.-Inform.(FH) **Roger Zacharias** ist Sun Certified Enterprise Architect und als Software Engineer bei der Wincor Nixdorf AG beschäftigt. Dort ist er für eine Software-Produktlinie verantwortlich, welche auf SOA- und EDA-Konzepten basiert. Als JCP-Mitglied beteiligt er sich zudem aktiv an der Weiterentwicklung der Standards und Konzepte im Java-Umfeld.

Links & Literatur

[1] OASIS: SOA Reference Model: www.oasis-open.org

[2] Zacharias, Roger: Serviceorientierung – Der OO-König ist tot, es lebe der SOA-König!, OBJEKTspektrum (2/2005)

[3] Luckham, David: The Power of Events – An Introduction to CEP in Distributed Enterprise Systems, Addison-Wesley (2002)

[4] Luckham, David: Complex Event Processing: www.complexevents.com

[5] ESP Community: Event Stream Processing: www.eventstreamprocessing.com

[6] Gartner: Technology Hypecycle 2006, Gartner (07/2006)