Børsen Ledelseshåndbøger er Danmarks største og stærkeste videns- og udviklingsklub. Uanset hvilket område eller emne du beskæftiger dig med, får du her et komplet opslagsværk på print, cd-rom og internet, der giver dig overblik og indsigt.

Ledelseshåndbogen er et praktisk og overskueligt værktøj til dig, der vil være 100% opdateret inden for et bestemt område – selvom du har en travl hverdag.

Børsen Ledelseshåndbøger

Artikel trykt i

E-business

Børsen Forum A/S • Børsen Ledelseshåndbøger • Møntergade 19 • DK-1140 København K
Tel: +45 33 32 01 02 • Fax: +45 33 91 08 30 • e-mail: handboger@borsen.dk • www.borsenhandboger.dk

11.3.

Event Driven Architecture

af Henrik Hvid Jensen, henrikhvid@soanetwork.dk

Virksomheder skal i dag være mere omstillingsparate end nogensinde før. Gartner definerer den b*ehændige virksomhed (RTE – realtime enterprise) som "den fuldstændige kompression af forsinkelse mellem at en hændelse opdages, rapportering af denne hændelse, beslutningen og svaret på hændelsen. For ens kunder, ledelsen og alle andre der har behov for information hurtigt for at træffe beslutninger, er det en drøm, der bliver til virkelighed".* Gartner estimerer også at, i 2006 vil de dominerende spillere i hvert forretningssegment have opnået førerskab eller forstærket deres førerskab gennem brug af reel tids-teknologi.

Vigtigste forretningshændelser ligger slumrende

Organisationer der prøver at blive behændige virksomheder, der opererer i reel tid, er ofte ikke opmærksomme på, at deres vigtigste forretningshændelser ligger slumrende, lukket inde i de sikre rammer af deres komplekse system. Hændelsesbaseret eller Event Driven Architecture (EDA) tilbyder muligheden for at åbne for forretningspotentialet i en verden, hvor forretningsprocesser og deres afhængighed af forretningshændelser sker uden forsinkelse.

Forestil en virksomhed hvor:

- Enhver vigtig forretningshændelse let, effektivt og troværdigt kan blive identificeret, opfanget og offentliggjort til virksomhedens "besked-bus" uden omkostningsfuld og risikabel applikationsombygning
- Informationen, der beskriver hændelsen, fortsætter gennem virksomheden med potentiale til at påvirke flere forretningsenheder, der kan udlede unik værdi ved at abonnere på disse hændelser
- Forretningsanalytiker kan dynamisk konfigurere hele systemet uden at iværksætte komplekse tekniske implementeringsopgaver.

Mange virksomheder har allerede afløst traditionelle forretningsprocesser med hændelsesbaserede processer. For eksempel:

- En virksomhed med en konventionel politik om at *"producere til lager"* forsøger at vedligeholde dets lagerbeholdning af færdige produkter indenfor en øvre og nedre grænse, bestemt af forudsigelser baseret på tidligere salg. En hændelses-drevet, *"producer til ordre"*-virksomhed afventer ankomsten af en kundeordre og initierer øjeblikkeligt produktion af den ønskede enhed.
- Flyselskaber benytter hændelses-baseret strategier for hyppigt at tilpasse priser baseret på kunders efterspørgsel *(dynamisk prissætning* i forhold til statisk prissætning). Indtjeningen er maksimeret ved at sætte en højere pris, hvis efterspørgslen er stærk og sætte en mindre for at fylde flere sæder, hvis efterspørgslen er svag. Det er kun muligt hvis software kontinuert følger kundernes billetindkøb (en hændelse) separat for hvert fly og dato.
- Ledelsen, transportplanlæggere og shippingmedarbejder bliver udstyret med browser-baserede grafisk "instrumentbræt", der opdateres ofte for at give synlighed i leverancekanalaktiviteterne. Det muliggør hurtigere reaktion på potentielle problemer og muligheder for at kapitalisere på hændelser hvis timing og type ikke kan forudses *(leverancekanalovervågning* i forhold til periodiske rapporter og ad-hoc forespørgsler).

Traditionelle processer er statiske og upræcise, fordi de påtvinger en forud planlagt sekvens af aktiviteter uafhængig af de øjeblikkelige betingelser. Hændelses-baserede processer reagerer på betingelser, som de er i virkeligheden, frem for hvordan de forventes at være. En hændelses-baseret tilgang kan ikke automatisk løse problemer, der fremkommer ved ekstreme betingelser, men det kan fange og rapporter undtagelser hurtigere end konventionelle systemer og derved give mere tid og data, som muliggør et bedre svar, når menneskelig involvering er nødvendig.

Et nyt begreb indenfor it-arkitektur er dukket op på det seneste og bliver ofte præsenteret som efterfølgeren til en Service Orienteret Arkitektur (SOA). Dette dokument vil give et overblik over hvad en Event Driven Arkitektur (EDA) indeholder og hvordan den passer ind i en SOA.

Det væsentligste budskab er, at EDA ikke er en separat arkitektur-tilgang, men er et væsentlig element i, hvordan en virksomhed bør implementere SOA korrekt.

SOA er en tilgang til distribueret computerbehandling, hvor softwarefunktionalitet tilbydes som services på et netværk. En Event Driven Arkitektur er en tilgang, hvor en hændelse/event afstedkommer, at en asynkron besked sendes mellem uafhængige software komponenter. Beskeden er envejs i forhold til service-interaktioner, hvor kommunikationen typisk er en tovejs anmod/svar kommunikation mellem en serviceforbruger og en serviceleverandør.

1. Hvad forstås ved event/hændelse?

Når begrebet event eller hændelse bruges, forstås en hændelse, der sker i den virkelige verden. Det vil sige noget, som ændrer tingenes tilstand fra et stadie til et andet. Det kan f.eks. være en forretningshændelse såsom en ordre, ankomsten af et parti vare til rampen eller betaling af en regning. Disse hændelser registreres typisk som en besked fra et stykke software til et andet og i mange tilfælde vil flere hændelser blive samlet til en ny samlet hændelse. Når der diskuteres it-arkitektur, er det typisk beskeden, der sendes mellem softwareenhederne, der betragtes som hændelsen.

Princippet i events har været kendt længe på et teknisk niveau indenfor systemsoftware, såsom operativsystemer og netværk- og systemhåndteringsværktøjer. Der er ikke så meget disse eventtyper, som dette dokument vil fokuserer på. Fokus vil være på hændelser, der sker i relation til organisationens forretningsområde.

I en EDA behøves de enkelte softwarekomponenter ikke at have noget kendskab til hinanden. En eventafsender sender typisk beskeden gennem noget middleware-integrationsmekanismer såsom en bus og denne middleware offentliggør beskeden til de objekter, som abonnerer på hændelsen. De vil derefter foretage sig nogle aktiviteter baseret på hændelsen. Selve hændelsen indkapsler en aktivitet og er en komplet beskrivelse af en specifik aktivitet. Når to deltagere i en hændelse ikke behøver at have nogen information om hinanden for at interagere, betragtes disse deltagere som *afkoblet.*

Kobling er et begreb der beskriver niveauet af fælles viden, som er nødvendig i en distribueret computer-udveksling. Der er tre begreber:

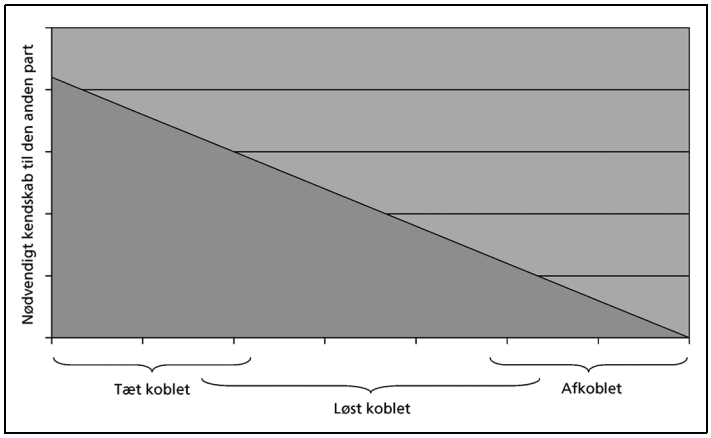
I en tæt koblet udveksling skal programmøren af den ene deltager have et *detaljeret kendskab* til opførslen af den anden deltager for succesfuldt at kunne udføre den krævede interaktion mellem to stykker software. Interaktionen vil typisk blive defineret på udviklingstidspunktet. Der vil typisk være behov et vist kendskab til, hvordan hver deltager er opbygget internt.

I en løs koblet udveksling skal deltagerne have *specifik men begrænset viden* om hinanden, typisk beskrevet i en WSDL-*servicekontrakt.* Servicekontrakten er et eksternt dokument, som leverer den information, hver deltagere behøver at vide for at kunne interagere med hinanden. Den eneste afhængighed mellem deltagerne er altså servicekontrakten og afhængigheden sker først på kørselstidspunktet ikke udviklingstidspunktet. På kørselstidspunktet henter forbrugeren servicekontrakten og baseret på indholdet i denne, dannes dynamisk en besked, der sendes til leverandøren. Serviceforbrugeren har intet kendskab til formatet af beskederne eller placeringen af servicen, før det skal bruge servicen. Der er intet behov for, at vide hvordan den anden part er opbygget internt.

I en afkoblet udveksling skal deltagerne *ikke have nogen viden* om hinanden for at kunne interagere. Det vil sige, at der er ikke behov for en servicekontrakt mellem deltagerne. Den eneste forbindelse mellem eventleverandøren og forbrugeren er offentliggør- og abonner-aktiviteterne. Der er derfor ikke tale om helt afkoblede udvekslinger, de skal have information om disse hændelser. Så afkobling betyder egentlig *mere løst koblet end kontraktbeskrevne services.* En *helt afkoblet* interaktion er en situation, hvor de to parter i interaktionen overhovedet ikke har nogen viden om strukturen eller indholdet af interaktionen, f.eks. i situationer hvor et stykke software offentliggøre en hændelse uden at levere nogen information om hændelsen og abonnenter på den hændelse kan modtage den og behandle dem, som de har lyst. En sådan interaktion vil typisk involvere mennesker til at forstå og fortolke hændelsen.

At kunne reagere dynamisk og fleksibelt på nye hændelser

Figur 1, Kobling er et flydende begreb, illustrerer, at kobling er et relativt begreb, hvor der ikke er klare afgrænsninger mellem hvornår man er tæt koblet, løst koblet og afkoblet. Man kan nærmere sige, at man er mere eller mindre koblet i forhold til andre distribuerede computerbehandlingsaktiviteter. Men generelt kan man sige, at jo mere man bevæger sig mod højre, jo større bliver it-infrastrukturens og derved virksomhedens behændighed. Men samtidig bevæger man sig væk fra de trygge forudsigelige rammer til en mere abstrakt situation med krav om at kunne reagere dynamisk og fleksibelt på nye hændelser. Der er ikke et ideelt niveau af løs kobling, det drejer sig primært om at bruge det rigtige værktøj til opgaven.

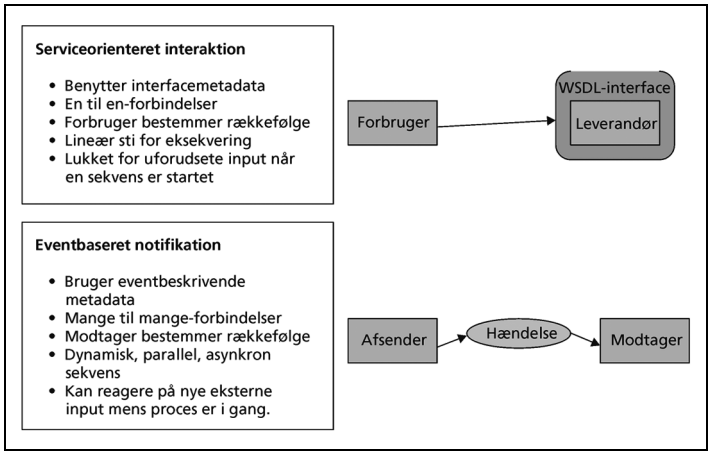


Figur 1. Kobling er et flydende begreb

og Figur 2, Illustration af serviceorienteret interaktion og hændelsesbaseret notifikation, sammenligner en serviceorienteret og eventbaseret kommunikation.

Tabel 1. Sammenligning mellem en hændelsesstyret- og servicestyret-kommunikation

Hændelse / event-baseret kommunikation	Serviceorienteret kommunikation
Udløseren er en forretningshændelse, der typisk sker på uforudsigelige tidspunkter	Processer er initieret under forudsigelige betingelser
Understøtter en til en, en til mange og mange til mange kommunikationer	En til en kommunikation
De resulterende aktiviteter bestemmes af modtageren baseret på den modtagne besked	Aktiviteter er kontrolleret af klienten (afsenderen)
Understøtter dynamiske, parallelle asynkrone arbejdsgange gennem et netværk af processer	Lineær eksekveringssti gennem et hierarki af services



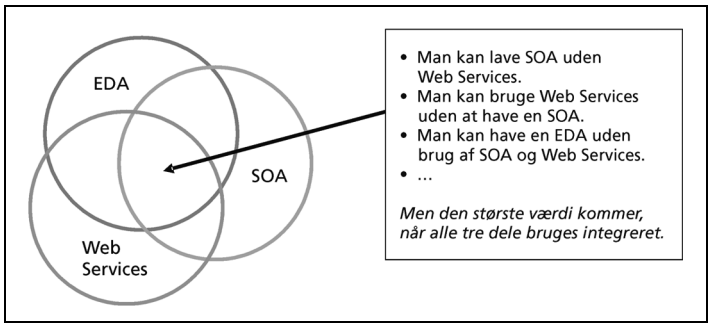
Figur 2. Illustration af serviceorienteret interaktion og hændelsesbaseret notifikation

2. Hvornår serviceorientering? Hvornår hændelsesbaseret?

Applikationsdesignere vil ofte mikse service og hændelsesrelationer i den samme applikation. En event-afsender behøver ikke at vide, hvem eller hvad der forventes at modtage eventen eller hvad modtageren vil gøre med det. Ligesom en objektorienteret udvikler ser på og definerer sine objekter, vil event-orienteret udviklere tænke på og definere deres hændelse som det primære fokusområde.

Et nyt mere virtuelt syn på virksomheden

Formålet for både SOA og EDA er at kombinere multiple moduler til store distribuerede applikationer, ved at separere interfacedefinitionen eller hændelsesbeskrivelsen fra software implementeringen. De kan begge benytte Web Services, men ingen kræver brug af Web Services (se dog Figur 3, SOA, EDA og Web Services giver størst værdi når de bruges sammen). I en SOA vil en forbruger finde, forbinde og derefter starte en service. I en event-baseret relation (notifikation) skal en afsender ikke finde, forbinde eller starte en modtager, den eneste kommunikation sker gennem data i eventen. Det tillader et nyt mere virtuelt syn på virksomheden, som spreder sig ud over de faste virksomhedsgrænser til at omfatte enhver anden forretningsenhed, som kunne blive berørt af hændelsen.



Figur 3. SOA, EDA og Web Services giver størst værdi når de bruges sammen

En serviceorienteret og eventbaseret applikation er forskellige på den måde, de strukturerer relationerne mellem modulerne, hvilket gør dem *brugbare til forskellige formål.* Services skal bruges, når forretningsproblemet, der skal løses, kræver en anmod/svar relation, hvor klientmodulet får nogle svar – øjeblikkeligt eller forsinket – fra en underordnet procedure, før det kan afslutte dets arbejde. Applikationer, der henter data fra nye og gamle ressourcer, passer fint med SOA. Ældre programmer kan blive indkapslet og moderniseret til at udstille deres funktionalitet gennem WSDL-interfaces og derefter tilgået fra nye kaldende applikationer. Multikanals og sammensatte applikationer er de bedste kandidater til serviceorientering.

Forretningskomponenter, som opererer uafhængigt

EDA er bedre egnet til at bygge autonome forretningskomponenter, som opererer uafhængigt. Virksomheder bør bruge hændelser til applikationer hvor multiple behandlingsstrømme kan eksekveres samtidigt; hvor timingen af

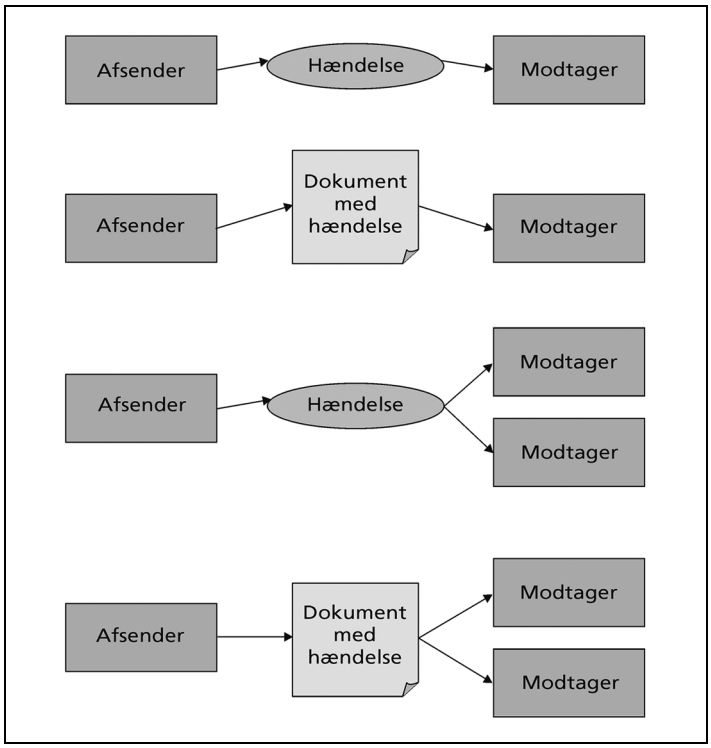
hændelsen, såsom begyndelse og afslutning af et trin er uforudsigeligt, eller ankomsten af yderligere eksternt input er uforudsigeligt; eller der er behov for dynamisk at tilføje, afbryde eller modificere procestrin uden at ændre andre kørende moduler. EDA forøger også forståelsen for, hvad der sker i en forretningsproces.

Event-baseret design vil blive en central komponent i forretningsarkitekturen og i it, fordi det giver muligheder for at understøtte dynamiske og mangesidet forretningsprocesser. Event vil komplementer ikke udskifte services i en serviceorienteret arkitektur. Det er gennem en kombination af serviceorientering og eventbaseret, at virksomheden vil opnå den ønskede behændighed.

3. Typer af hændelser

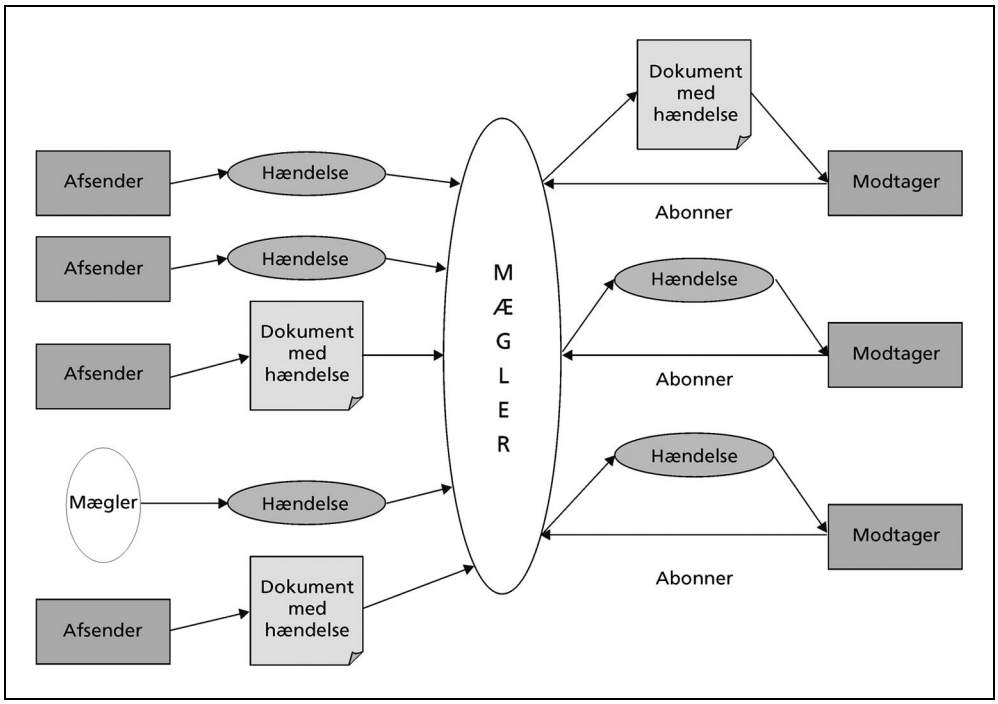
Event-baserede forretningsapplikationer kan inddeles i fire kategorier:

Simple event-baserede (eller beskedbaserede) applikationer, hvor applikationer eksplicit sender eller modtager en besked direkte til eller fra hinanden. F.eks. gennem Web Services eller beskedorienteret middleware (BOM/MOM). Beskeden betragtes kun som en hændelse, hvis det afspejler fremkomsten af en ordinær hændelse. Der vil være grænsetilfælde, mellem hvornår det er en besked og hvornår det er en hændelse. F.eks. et XML dokument med en indkøbsordre, låneansøgning eller lignende transaktion kan betragtes som en slags hændelse, hvis de kun sendes en vej, da de jo afspejler en forretningshændelse. De er mindre rene end en klart designet hændelse, der rapporterer en enkel ændring af tilstand. Et dokument bliver sendt fra en applikation til en anden, så der er et løbende log af tilstandsændringer. Hvis dokumentet bruges i et to-vejs anmod/svar-mønster, er det *ikke* en hændelse.



Figur 4. Illustration af simple hændelser

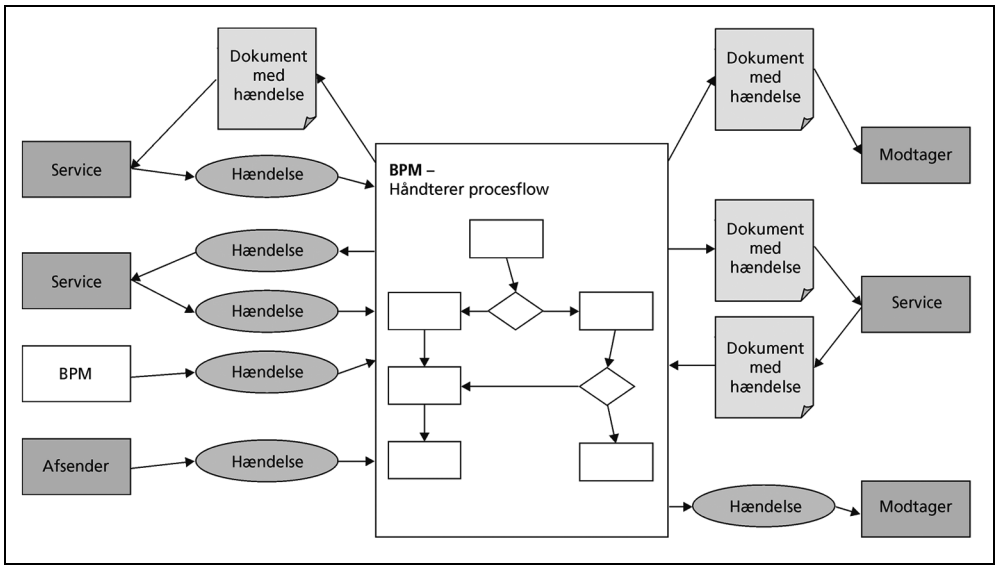
Event-baserede applikationer med en integrationsmægler som mellemled. Integrationsmægleren kan lave syntaktiske og semantiske transformationer og sender simple hændelsesbeskeder i henhold til nogle logiske regler. Den bruger ofte offentliggør abonner-kommunikation. Mæglerne tilfører altså noget "intelligens" ved brug af regler, dette i sammenligning med traditionelle BOM som kun understøtter basale hændelsesleverancer. Brugen af et mellemled er et væsentligt trin i forhold til simpel hændelsesbaseret applikationer, fordi en del af beslutningerne om dataflow og kontrolflow er flyttet fra applikationen og ind i mægleren. En mægler vil også understøtte en SOA, så den skal ikke kun ses som værktøjer til at håndtere hændelser.



Figur 5. Illustration af mægler

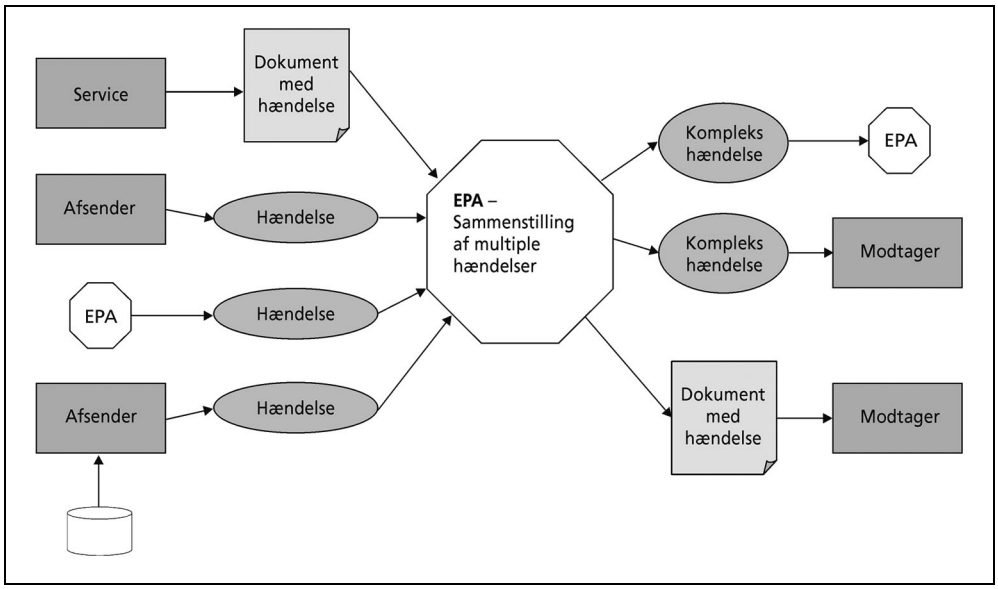
Håndterer hele processen ved brug af specielle typer af event

Event-baserede applikationer der er dirigeret af en forretningsproceshåndteringsenhed (Business Process Management BPM). Den håndterer hele processen ved brug af specielle BPM-orienterede typer af event. BPM håndtere procesflowet ved at "huske" status for hver instans af processen. Den bruger regler til at evaluere indhold og hændelser og iværksætter udførslen af det næste hensigtsmæssige trin. Den bruger ofte specielle skræddersyede BPM hændelsesbeskeder til at kommunikere mellem en BPM adapter og en central proces for at markere start og slut af en aktivitet. Disse specielle hændelser understøtter koordinationen af BPM processen, men er ikke forretningshændelser. De bærer ikke applikationsdata og er ikke sendt eller modtaget af et applikationsprogram. BPM værktøjerne bruger sådanne hændelser til at opsplitte og samle kontrolflow baseret på logiske regler.



Figur 6. Illustration af BPM

Komplekse hændelsesbehandlings-applikationer (Complex-event processing CEP). Hvor en sofistikeret hændelsesansvarlig enhed logisk evaluerer multiple hændelser for at muliggøre afkoblede, parallelle, asynkrone applikationsbehandlinger. CEP bruger en central hændelsesansvarlig eller et antal distribuerede hændelses-behandlings-agenter (EPA), som bruger regler til at filtrere hændelser, sammenstille en mængde af hændelser til et højere niveau af komplekse hændelser ved at sammenholde hændelserne med mønstre og finde hændelsesmønstre, der ikke overholder visse betingelser. En hændelsesansvarlig kan danne nye hændelser, som sendes til applikationer eller til andre hændelsesansvarlige.



Figur 7. Illustration af CEP

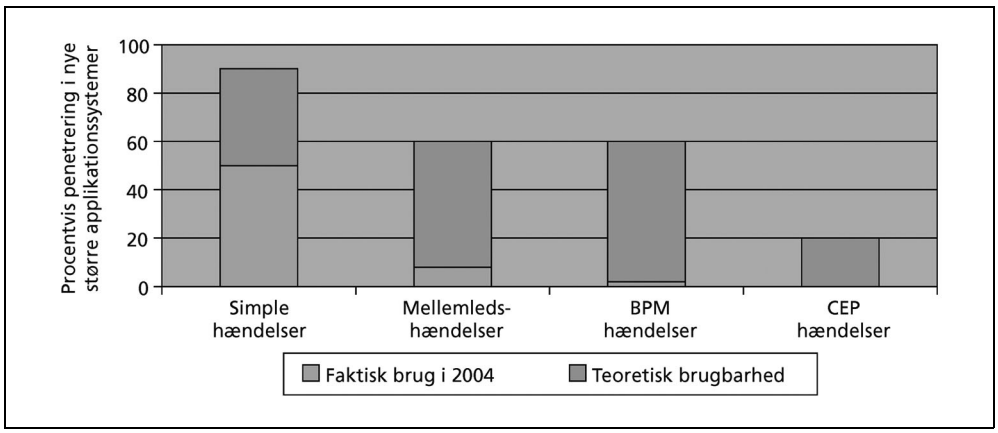
I mange miljøer med multiple adskilte informationssystemer, vil en væsentlig hændelse kun blive opdaget ved at evaluere, sammenligne og sammenstille nye informationer eller ændringer med data tilgængelig fra andre kilder. For eksempel når et skib kommer i havnen, behøves det ikke være en hændelse af nok betydning til at udstede en "alarm", men hvis andre systemer viser, at en interessant person tidligere har været en del af skibets besætning, kan en mere dybdegående undersøgelse være på sin plads. I denne situation er hændelsen at et nyt skib er ankommet placeret i et system, mens informationen om den interessante person er placeret i et andet. En CEP tillader virksomheder at udnytte multiple informationskilder som del af en enkelt proces, sådan at data fra multiple systemer kan bruges i reel tid til at evaluere hvilket svar, der skal ske baseret på en ny hændelse. Denne beskedforædling er et vigtigt krav, da faktorer tilstede på det tidspunkt hvor hændelsen sker, kan levere essentiel kontekst forståelse for at gøre hændelsen brugbar.

Et andet eksempel på en ikke planlagt forretningshændelse er, når indtjeningen falder regionalt. Reaktionen på denne hændelse kan være at starte en service, der kan undersøge hændelsen (f.eks. bestemme om den lokale leverandørs lager er tomt, om efterspørgslen er faldet, om medarbejderomkostningerne er steget) og derefter kalde en anden ser-

vice, som adresserer problemet (indhenter varer fra en anden leverandør, lancerer en marketingkampagne eller starter produktion hos tredjepart).

Projektstyrings-systemerne separeret fra logistiksystemer

I mange produktionsmiljøer er projektstyringssystemerne separeret fra logistiksystemer, der holder rede på leverance af aktiver, der influere på projektets afslutning. CEP leverer muligheden for, at integrere adskilte systemer, der indeholder data, der konstant skal sammenholdes. F.eks. kan en projektleder specificere en opgave i projektstyringssystemet (f.eks. at bygge en sektion af et fly på en bestemt dato). For at denne sektion af flyet kan afsluttes som skemalagt, skal aktiver, som følges i totalt separerede logistik systemer, leveres til tiden. I mange situationer vil en person få til opgave at sammenstille hændelser i logistiksystemet med opgaver i projektstyringssystemet. Ved at placere en CEP ovenpå begge systemer vil alle hændelser blive håndteret automatisk. Brugere kan tilgå et enkelt browserbaseret interface for at se hændelser og data som sker i de enkelte systemer og hvis et konkret aktiv ikke leveres til tiden, kan CEP beslutte, om den skal advisere den ansvarlige, baseret på hvordan den forventede leverancedato influerer på det samlede projekt.



Figur 8. Brug af Event Driven Architecture (kilde Gartner)

Gartner har i Figur 8, Brug af Event Driven Architecture (kilde Gartner), vurderet hvor stor en del af behandlingen, der håndteres af de enkelte hændelsesformer. Gartner vurderer, at brugen af Enterprise Service Bus (ESB) vil understøtte serviceorienteret og eventbaseret design i en brugervenlig, standardbaseret middleware infrastruktur. Den stigende brug af BPM værktøjer vil forøge brugen af mere

kraftfulde former for hændelser, mens væksten i brugen af CFP vil være mere langsom, dels fordi mulighederne er mere begrænsede og dels fordi det kræver væsentlig større ekspertise at udvikle løsninger og gøre brug af resultatet.

4. Hændelsesbaseret integration

De fleste integrationsinitiativer har været pull eller anmod/svar orienterede. Det betyder at timing af integrationsprocessen sker under kontrol af selve integrationsplatformen. Integrationsplatformen anmoder eller trækker information fra underliggende applikationssystemer for at have information tilgængelig. Denne udtrækningsproces sker på foruddefinerede intervaller. Sådanne initiativer har tilført en del forretningsværdi, men er kun et trin for at gøre organisationen mere behændig. For at reducere friktionen inde i forretningsprocessen må forretnings- og integrationsprocesser ofte udløses af hændelser, der sker i det underliggende system i reel tid. En hændelsesbaseret integrationsstrategi kan leverer konsistent information på tværs af virksomheden hurtigt. Ofte er hændelsesdrevet integration den eneste måde at sikre konsistens, da mange hændelser kun giver mening, hvis de er opfanget præcis på det tidspunkt, hvor de skete.

I mange virksomheder sker større forsinkelser

I mange virksomheder sker der større forsinkelser mellem at en hændelse sker og den er blevet identificeret af afdelingen, der skal håndtere den. Afhængig af typen af en hændelse og dets vigtighed kan værdien af en hændelse formindskes proportionalt med den tid, der er gået, siden den skete. For visse hændelser er alt andet end øjeblikkelig behandling uacceptabelt. For andre kan timer eller dage passere uden væsentlig forringelse af forretningsværdien. Hændelsesbaseret integration er essentiel for den første og kan repræsentere interessante nye forretningsmuligheder for den sidste. Hændelsesbaseret integration skal derfor fokusere på, hurtig at få informationen der beskriver hændelsen hen til modtageren. Det skal i hvert fald ske så hurtigt, at værdien af hændelsen ikke degraderes.

5. Hændelsesbaseret design

Hændelses-baseret design er en mulighed for at forøge den behændige og allestedsnærværende computerbehandling.

Hændelser er baseret på notifikationer, en afkoblet type af kommunikation der er mere skalerbar og fleksible end tæt koblede eller løst koblede interaktioner.

Hændelser tilbyder to yderligere former for behændighed:

Instansbehændighed – Hver instans af en forretningsproces behandles individuelt, såsom at sætte forskellige enheder og farver på hver enkelt bil på samlebåndet

Procesbehændighed – Modificerer processen som svar på ændringer i verden. Såsom at ændrer samlebåndet til at bygge MPV i stedet for sedan.

Selvom eventbaserede forretningsprocesser primært er et forretningsanliggende, har de implikationer for it-afdelingen. Mange aspekter af hændelses-baserede forretningsstrategier kan kun blive implementeret, hvis de underliggende applikationssystemer også er eventbaserede. Traditionel applikationsarkitektur er for statisk, ufleksibel, for svære at skalere og uhåndterlig til at implementere den type af proaktiv opførsel, som hændelses-baserede processer behøver. Event-baserede applikationer tillader processer at blive modificeret hurtigt og reagere på fejl og exceptionelle hændelser, der forstyrre konventionelle processer.

6. Forventninger

Virksomheder må forberede sig på, at alle typer af applikationer vil udsende en kontinuer strøm af beskeder, hver gang der sker en væsentlig aktivitet. Gartner beskriver en situation, hvor der udover disse strømme af hændelser også vil være muligt rutinemæssigt at skanne dokumenter, filer og beskeder, der strømmer gennem virksomhedens netværk, for meningsfyldt information. At der vil være opstillede strategisk placerede sensorer, der observerer hændelser, der sker i den fysiske verden udenfor it. Alle disse hændelser vil blive opsamlet og der vil være adgang til disse informationer for mennesker og for maskiner, der begge kan reagere hensigtsmæssigt baseret på mønstre i disse hændelser. For at opnå et sådant niveau af forståelse er det nødvendigt at implementere simple metoder til opsamling af eventdata, intelligente agenter må installeres i netværket for at evaluere eventbeskeder og afvise dem, som der ikke er behov for og opsummere lavniveau observationer til højere niveauer af komplekse hændelser passende til at blive præsenteret for

mennesker eller som afstedkommer automatiske opfølgningsaktiviteter. Semantic Web standarderne bliver ikke direkte nævnt i litteraturen omkring EDA, men Semantic Web tankegangen vil være én væsentlig komponent i en sådan situation.

Figur 9. Hvordan hændelsesbaserede forretningskrav håndteres af hændelsesbaserede applikationer

Hændelsesbaserede forretningsproceskrav	Løses af hændelsesbaseret applikationer ved
Reducere den samlede tid en samlet proces tager	• Håndtere hændelser individuelt, frem for samlet • Kører aktiviteter parallelt frem for serielt • Bruger *push* ikke *pull* kommunikation • Notificerer mennesker selektivt
Behandler hver instans af en forretningsproces individuelt (instans behændighed) eller modificerer selve processen (proces behændighed)	• Trækker forretningsregler ud af applikationsprogrammer • Data- og procesflow-beslutninger er foretaget hos mæglere, BPM løsninger og andre regelbaserede løsninger
Reagerer på undtagelser og andre nye hændelser kontinuert gennem hver enkelt forretningsprocesinstans livscyklus	• Modulært design • Dynamisk arkitektur • Håndtering af komplekse hændelser

7. Industristandarder

Industristandarderne for hændelsesbaserede applikationer er ufuldstændige, men efterhånden som de modnes, vil de forøge brugen af hændelseskonceptet og tilbyde konsistent hændelseshåndtering og forståelse mellem softwareprodukter. I øjeblikket er de væsentligste standarder, der er under udvikling WS-Eventing og WS-Notification, disse to specifikationer vil blive samlet under WS-Eventing. Den er en standard for asynkron notifikation af hændelser mellem XML-baserede Web Services. Det er et antal protokoller, beskedformater og interface, der tillader Web Services at abonnere eller acceptere abonnenter til hændelsesnotifikation. Derudover er standarder, som f.eks. WS-Reliable Messaging til at sikre fælles forståelse for status for leverance af hændelsen og WS-Adressing der kan bruges til at under-

støtte nogle af de opgaver, som kræves af en mægler, væsentlige indenfor en EDA.

Det bliver også en opgave for diverse industriorganisationer at definere hændelser indenfor deres industrisegment, så der kan være en fælles forståelse for hændelsernes betydning indenfor deres domæne. Semantic Web standarderne vil være velegnede til at beskrive de enkelte hændelser.

8. WS-Notification

Web Services Notification er en familie af specifikationer, der definerer en standard Web Services tilgang til notifikation ved brug af en emne/topic-baseret offentliggør/abonner mønster. Det inkluderer en standard beskedudveksling, der skal implementeres af serviceleverandøren, der ønsker at deltage i notifikationer, standard beskedudveksling for en leverandør af notifikationsmæglerservice (det tillader offentliggørelse af beskeder fra enheder, der ikke selv er serviceleverandører). En XML model der beskriver emner og operationelle krav til serviceleverandør og anmoder der deltager i notifikationen.

9. WS-Eventing

Denne specifikation definerer en protokol for hvordan en Web Service kaldet en "event sink" kan registrere interesse (abonnerer) i en anden Web Service og modtage beskeder om hændelser.

10. Opsummering

Hændelsesbaserede processer er mere informationsintensive end konventionelle processer, fordi hændelser bliver genereret for at markere ting, som normalt ikke ville være blevet registreret.

Applikationsudviklere har traditionelt ikke set et behov for øjeblikkeligt at transmittere en hændelsesbesked for hver ændring af tilstand bare for at forøge synligheden af processen eller accelerere aktiviteterne hos andre forretningsenheder eller applikationer. Men den teknologiske udvikling både i form at hurtigere, billigere og mere stabile netværk

samt den generelle tilgængelighed af nye teknologier såsom radio-frequency identification mærkater (RFID), gør det praktisk at opsamle og udsende hændelsesinformation mere bredt end tidligere.

Der er finansielle og strategiske fordele ved at implementere hændelses-baserede forretningsprocesser, fordi de nedarver den hændelsesbaserede natur af mange aspekter i den virkelige verden.

Hændelsesbaserede applikationer tillader processer at blive modificeret hurtigt og hurtigt reagere på fejl og exceptionelle betingelser, som forstyrrer konventionelle processer.

Generelle anbefalinger:

- Design alle nye virksomhedsapplikationer som et antal tilgængelige moduler frem for med en monolitisk arkitektur
- Brug et centralt integrationskompetencecenter til at vedligeholde dokumentation for SOA interfaces og hændelsesbeskrivelser
- Brug SOA og de tre simple former for hændelsesdesign i almindelige projekter

Event Driven Architecture tilbyder yderligere værdi til eksisterende it-systemer ved at åbenbare tidligere gemte hændelser, der kan udløse værdifulde, tværgående forretningsprocesser i reel tid.

I sammenligning med traditionel pull (svar/anmod) informationsflow, vil hændelsesbaseret integration tilbyde en mere effektiv måde at identificere forretningshændelser og accelerere initieringen af hændelsesafhængige forretningsprocesser.

Serviceorientering og eventhåndtering bør ikke betragtes som to separate koncepter, men mere som en måde at tilvejebringe værdi til organisationen under en samlet SOA-banner.